Learning Classifiers from Semantically Heterogeneous Data

Doina Caragea, Jyotishman Pathak, and Vasant G Honavar

Artificial Intelligence Research Laboratory
Department of Computer Science
Iowa State University
Ames, IA 50011-1040, USA
{dcaragea, jpathak, honavar}@cs.iastate.edu

Abstract. Semantically heterogeneous and distributed data sources are quite common in several application domains such as bioinformatics and security informatics. In such a setting, each data source has an associated ontology. Different users or applications need to be able to query such data sources for statistics of interest (e.g., statistics needed to learn a predictive model from data). Because no single ontology meets the needs of all applications or users in every context, or for that matter, even a single user in different contexts, there is a need for principled approaches to acquiring statistics from semantically heterogeneous data. In this paper, we introduce ontology-extended data sources and define a user perspective consisting of an ontology and a set of interoperation constraints between data source ontologies and the user ontology. We show how these constraints can be used to derive mappings from source ontologies to the user ontology. We observe that most of the learning algorithms use only certain statistics computed from data in the process of generating the hypothesis that they output. We show how the ontology mappings can be used to answer statistical queries needed by algorithms for learning classifiers from data viewed from a certain user perspective.

1 Introduction

Recent advances in computing, communications, and digital storage technologies, together with development of high throughput data acquisition technologies have made it possible to gather and store large volumes of data in digital form. For example, 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 in biological sciences; organizations have begun to capture and store a variety of data about various aspects of their operations (e.g., products, customers, and transactions); complex distributed systems (e.g., computer systems, communication networks, power systems) are equipped with sensors and measurement devices that gather and store a variety of data for use in monitoring, controlling, and improving the operation of such systems.

These developments have resulted in unprecedented opportunities for large-scale data-driven knowledge acquisition with the potential for fundamental gains in scientific understanding (e.g., characterization of macro-molecular structure-function relationships in biology) in many data-rich domains. To exploit these opportunities scientists at different institutions need to collaborate and share information and findings in a field or across various research fields [1]. Thus, researchers working at one level of a problem may benefit from data or results developed for a different level of that problem or even for a different problem.

However, more often than not, it is not easy for a scientist to be able to use information obtained from a different scientific community. Furthermore, even scientists working on the same problem at different institutions find it difficult to combine their results. These difficulties arise because of the large volume of information that would need to be moved around or because of privacy considerations. Even in cases when data can be shared, there are difficulties coming from the heterogeneity of the data collected by different scientific communities or organizations. This heterogeneity could be in terms of structure (relational databases, flat files, etc.) or content (different ontological commitments, which means different assumptions concerning the objects that exist in the world, the properties or attributes of the objects, the possible values of attributes, and their intended meaning) [2].

Against this background, we consider the problem of data driven knowledge acquisition from autonomous, distributed, semantically heterogeneous data sources [3]. Our approach to this problem comes from revisiting the traditional formulation of the problem of learning from data and observing that most of the learning algorithms use only certain *statistics* computed from the data in the process of generating the hypotheses that they output. [1] This observation yields a natural decomposition of a learning algorithm into two components: an *information extraction* component that formulates and sends a statistical query to a data source and a *hypothesis generation* component that uses the resulting statistic to modify a partially constructed hypothesis (and further invokes the information extraction component as needed). The information extraction from distributed data entails decomposing each statistical query q posed by the information extraction component of the learner into sub-queries $q_1, \cdots, q_K$ that can be answered by the individual data sources $D_1, \cdots, D_K$, respectively, and a procedure for combining the answers to the sub-queries into an answer to the original query q. In addition to that, in order to be able to use machine learning approaches to acquire knowledge from semantically heterogeneous data, a variant of the problem of information integration [2] needs to be solved.

The work described in this paper extends current approaches to information integration [2] and our previous work on learning from distributed data to develop principled methods for learning classifiers from semantically heterogeneous data [4]. This is achieved by associating an ontology with each data source and

[1] In general, a *statistic* is simply a function of data and any kind of query that returns such a statistic is called a *statistical query*. Examples of statistics include counts of instances that have specified values for a subset of attributes, called *join counts*.

thus, reducing the problem of learning from heterogeneous data to the problem of developing sound techniques for answering statistical queries from semantically heterogeneous data sources (see Figure 1).

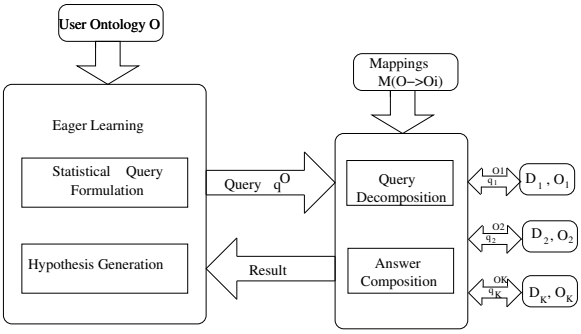

Fig. 1. Learning from Semantically Heterogeneous Distributed Data: each data source has an associated ontology and the user provides a global ontology and mappings from the local ontologies to the global ontology.

In the rest of the paper we identify *sufficient statistics* for a class of learning algorithms and show how we can gather these statistics from semantically heterogeneous data sources from a certain user perspective. To do that we define *ontology-extended data sources* and *interoperation constraints* between ontologies and present a way to automatically infer ontology mappings from the set of interoperation constraints specified by a user. We show how these mapping can be used to gather sufficient statistics. We demonstrate our approach using the Naive Bayes (NB) algorithm.

2 Statistics for Learning From Data

In a distributed setting, the data are distributed over data sources $D_1, \cdots, D_K$, where each data source contains only a fragment of the whole data. If the data is also semantically heterogeneous, each data source D_i has an associated ontology O_i. We assume that a user who wants to use some of the data available in the system for learning classifiers has also an associated ontology O_U.

Definition: The problem of *learning from semantically heterogeneous data* can be defined as follows: given the distributed, semantically heterogeneous data sources $D_1, \cdots, D_K$ with the associated ontologies $O_1, \cdots, O_K$ and a user ontology O_U, a hypothesis class H and a performance criterion P, the task of the learner L is to output a hypothesis $h \in H$ that optimizes P by integrating the data sources $D_1, \cdots, D_K$ according to the user ontology O_U.

Our approach to the problem of learning from semantically heterogeneous data relies on sufficient statistics.

Definition [5]: A statistic $s(D)$ is called a *sufficient statistic* for a parameter θ if $s(D)$ (loosely speaking) provides all the information needed for estimating the parameter θ from data D. Thus, sample mean is a sufficient statistic for mean of a Gaussian distribution.

We can generalize this notion of a sufficient statistic for a parameter θ to yield the notion of a sufficient statistic $s_L(D, h)$ for learning a hypothesis h using a learning algorithm L applied to a data set D [4]. Trivially, the data D is a sufficient statistic for learning the hypothesis h using L applied to D. However, we are typically interested in statistics that are minimal or at the very least, substantially smaller in size than the whole data set D.

We observed that a large class of learning algorithms such as Naive Bayes [6], Bayesian Networks [7,8], Bags of Words [6], Decision Trees [9], Relational Learning [10, 11], NB-k [12], Association Rules [13] etc. need only sufficient statistics of type *join count* computed from the data in the process of generating a hypothesis.

For some learning algorithms the sufficient statistics needed to generate a hypothesis can be computed in one step (e.g., Naive Bayes), while for others it is necessary to interleave statistics gathering and hypothesis generation (e.g., Decision Tree learning algorithm would first obtain the sufficient statistics for a partial hypothesis h_1 consisting of a single node, then follow up with queries for additional statistics needed to iteratively refine h_1 to obtain a succession of partial hypotheses h_1, h_2, $\cdots$ culminating in h, the final decision tree).

Naive Bayes Classifier

Learning Phase:
For each class c_j and each attribute value a_i compute the probabilities $P(c_j)$ and $P(a_i|c_j)$ based on their frequencies over the training data.

Classification Phase:
Given a new instance $\mathbf{x} =< a_1, \cdots, a_n >$ to be classified

Return $c_{NB}(\mathbf{x}) = \arg\max_{c_j \in C} P(c_j) \prod_{i=1}^{n} P(a_i|c_j)$

Fig. 2. Naive Bayes Algorithm

We will illustrate our approach to the problem of learning from semantically heterogeneous data using the Naive Bayes algorithm as an example.

2.1 Sufficient Statistics for Naive Bayes algorithm

In Naive Bayes framework (Figure 2), each example $\mathbf{x}$ is described by a conjunction of attribute values, i.e. $\mathbf{x} =< a_1, \cdots, a_n >$. The class label of an example can take any value from a finite set $C = \{c_1, \cdots, c_m\}$. We assume that the attribute

values are conditionally independent given the class label. A training set of labeled examples $D = \{< \mathbf{x_1}, y_1 >, \cdots, < \mathbf{x_t}, y_t >\}$ is presented to the algorithm. During the learning phase, a hypothesis h, represented as a set of probabilities $P(c_j)$ and $P(a_i|c_j)$, is learned from the training set. During the evaluation phase, the learner is asked to predict the classification of new instances $\mathbf{x}$. The set of probabilities $P(c_j)$ and $P(a_i|c_j)$, representing the hypothesis, can be computed based on counts of the form $t_j = count_D(c_j)$, and $t_{ij} = count_D(a_i|c_j)$. Thus, these counts represent sufficient statistics for the hypothesis build during the learning phase of Naive Bayes classifiers and can be computed in one pass through the data.

The Naive Bayes algorithm for learning from data can be easily extended to yield an algorithm for learning from horizontally distributed data by computing the counts at the distributed data sources and combining them at a central location to give a global count.

3 Answering Statistical Queries From Ontology-Extended Data Sources

In order to learn classifiers from semantically heterogeneous distributed data, techniques need to be developed for answering statistical queries, posed by the learner in terms the user ontology O_U, from the heterogeneous data sources. To achieve this we introduce the notion of *ontology-extended data sources*, which allows us to perform sound information integration. Our model is inspired from a similar model called *ontology-extended relational algebra* described in [14]. Although we can view a collection of physically distributed, autonomous, heterogeneous data sources as *though* they were relational databases [3], we will use the term *data sources* and not *relational databases* in what follows, to point out that, in principle, our data sources can be any kind of data sources (e.g., flat files, relational databases, web pages etc.). We will explain the concepts in this section using the following example.

3.1 Example

Suppose a company C_1 records information about weather in some region of interest R. From C_1's point of view, *Weather* is described by the attributes *Temperature*, *Wind*, *Humidity* and *Outlook*. An ontology O_1 associated with this data could tell us that *WindSpeed* is part of the *Wind* attribute description (called *part-of* relationship) and that *Sunny*, *Rainy*, *Cloudy* and *Snowy* are all *Outlook* descriptions (called *is-a* relationship). It can also tell us that the *Temperature* is measured in *degrees Fahrenheit* and the *WindSpeed* is measured in *miles per hour*. The data D_1 that this company collects can be stored in a table as shown in Table 1.

Suppose that another company C_2 collects information about weather in the same region R. From C_2's point of view *Weather* is described by the attributes temperature denoted *Temp*, *Wind*, *Humidity* and precipitations denoted *Prec*. The ontology O_2 associated with its data tells us that *Speed* and *Direction* are

Table 1. Data set D_1: Weather Data collected by company C_1

Day	*Temperature*	*WindSpeed*	*Humidity*	*Outlook*
1	20	16	67	*Cloudy*
2	10	34	53	*Sunny*
3	17	25	62	*Rainy*

both parts of the *Wind* attribute (*part-of* relationship) and that *Snow*, *Rain* and *NoPrec* are both *Prec* (*is-a* relationship). This ontology also stores information about the amount of precipitation by quantifying the precipitation values. For example, when recording the precipitation for one day, one can say *Rain* or *LightRain* or *HeavyRain* etc. (so *LightRain* *is-a* description of *Rain*). Furthermore, the ontology tells us that *Temp* is measured in *degrees Celsius* and that *Speed* is measured in *kilometers per hour*. Thus, the data D_2 collected by this company looks like the one shown in the Table 2.

Table 2. Data set D_2: Weather Data collected by the company C_2

Day	*Temp*	*WindSp*	*WindDir*	*Humidity*	*Prec*
1	3	24	N	67	*Rain*
2	-2	50	NW	53	*LightRain*
3	0	34	NE	62	*NoPrec*

Suppose that a user U, having his or her own semantic about the weather domain, wants to infer some global information about weather in region R using the data collected by both C_1 and C_2. Assume that in this user ontology O_U, *Temperature* (measured in degrees Fahrenheit), *Wind* described by *WindSpeed* (measured in mph) and *WindDir*, *Humidity* and *Precipitations* are the significant attributes. In order to be able to use simultaneously both data sources D_1 and D_2, the user needs to specify mappings from the data source ontologies O_1 and O_2 to his ontology O_U. For example, the user would map *Temperature* in O_1 and *Temp* in O_2 to *Temperature* in O_U ontology. The user needs also to specify a conversion function to convert *Temp* values in O_2 from degrees Celsius to Fahrenheit. Similarly, the user defines mappings and conversion functions for *WindSpeed*. With respect to *Precipitations*, the user observes that *Outlook* in O_1 and *Prec* in O_2 can be mapped to *Precipitations* in O_U. Also *Rainy* in O_1 can be mapped to *Rain* in O_U etc. In principle, a different user U' with a different semantic (ontology $O_{U'}$) may also want to use the data sources D_1 and D_2 for weather analysis. Similar to the first user, this user needs to specify mapping and conversion functions from the data source ontologies to his or her own ontology. Thus, every user can use the available data sources from his or her own perspective.

3.2 Ontologies and Mappings

Having the above example in mind, we will formally define the terms used, by extending the definitions in [14] from relational databases to general data sources (represented as tables).

Definition [14]: Let S be a partially ordered set under the ordering $\leq$. We say that an ordering $\preceq$ defines a *hierarchy* on S if the following three conditions are satisfied:

- $x \preceq y \Rightarrow x \leq y,\ \forall x, y \in S$ (we say that $(S, \preceq)$ is *more concise than* $(S, \leq)$),
- $(S, \leq)$ is the reflexive, transitive closure of $(S, \preceq)$,
- no other ordering $\sqsubseteq$, which is more concise than $(S, \leq)$, satisfies the above two conditions.

Example: Let $S = \{Weather, Wind, WindSpeed\}$. We can define a partial ordering $\leq$ on S according to the *part-of* relationship. Thus, *Wind* is *part-of* the *Weather* description, *WindSpeed* is also *part-of* the *Weather* description, and *WindSpeed* is *part-of Wind* description. Besides, everything is *part-of* itself. Therefore, $(S, \leq)$ = {(*Weather, Weather*), (*Wind, Wind*), (*WindSpeed, WindSpeed*), (*Wind, Weather*), (*WindSpeed, Weather*), (*WindSpeed, Wind*)}. It follows that $(S, \preceq)$ = {(*Wind, Weather*), (*WindSpeed, Wind*)} is the only one hierarchy associated with the order determined by the *part-of* relationship. Furthermore, $(S, \leq)$ is the reflexive, transitive closure of $(S, \preceq)$.

Let Λ be a finite set of strings that can be used to define hierarchies for a set of terms S. For example, Λ may contain strings like *is-a, part-of* corresponding to *is-a* and *part-of* relationships, respectively.

Definition [14]: An *ontology* O (over terms in S) with respect to the partial orderings contained in Λ is a mapping Θ from Λ to hierarchies on S defined according to orderings in Λ.

In other words, an ontology associates orderings to their corresponding hierarchies. Thus, if *is-a* $\in \Lambda$, then Θ*(is-a)* will be the *is-a* hierarchy associated with the set of terms in S. For example, Figures 3, 4 and 5 show the ontologies

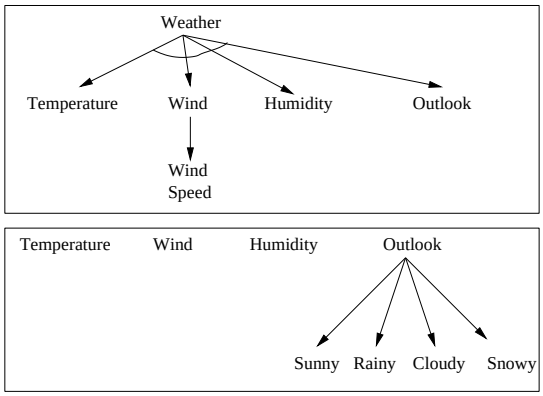

Fig. 3. Ontology O_1 associated with the data source D_1

associated with the data sets D_1 and D_2, and the user ontology O_U, respectively, when $\Lambda = \{$*is-a, part-of*$\}$. In this case, the ontologies consist of *is-a* and *part-of*

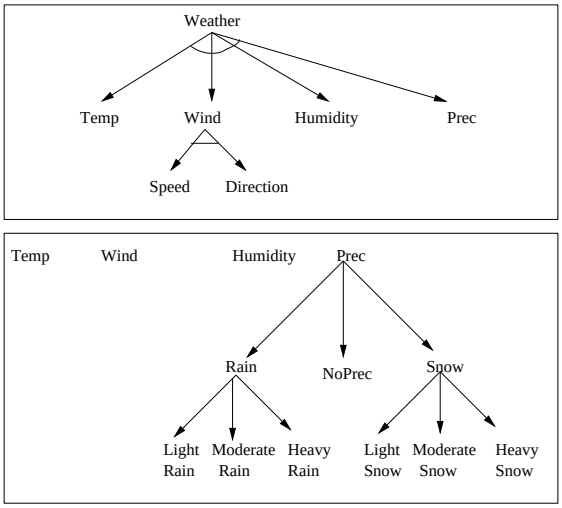

Fig. 4. Ontology O_2 associated with the data source D_2

hierarchies.

As mentioned before, we want to associate ontologies $O_1, \cdots, O_K$ with distributed data sources $D_1, \cdots, D_K$. For a user having an ontology O_U to be able to ask queries over several autonomous heterogeneous data sources, the user needs to specify mappings from the data source ontologies $O_1, \cdots, O_K$ to the user ontology O_U, so that all the ontologies $O_1, \cdots, O_K$ are integrated according to the ontology O_U.

Definition [14,15]: Let $(H_1, \preceq_1), \cdots, (H_K, \preceq_K)$ be a set of K hierarchies determined by the same relationship *ord* (e.g., *is-a*) on the sets of terms $S_1, \cdots, S_K$, respectively, and let $(H_U, \preceq_U)$ be a user ontology determined by the relationship *ord* on a set of terms S. A *set of interoperation constraints IC(ord)* is a set of relationships that exist between elements from hierarchies H_i and elements from the hierarchy H_U. Thus, for two elements $x \in H_i$ and $y \in H_U$ we can have one of the following IC's - $x : H_i = y : H_U$ or $x : H_i \neq y : H_U$ or $x : H_i \leq y : H_U$ or $x : H_i \not\leq y : H_U$.

Example: For the weather example, if we consider the *is-a* hierarchies associated with the data sources D_1 and D_2 (i.e., H_1*(is-a)* and H_2*(is-a)*) and the *is-a* hierarchy H_U*(is-a)*, we have the following interoperation constraints, among others: $Temp : H_2$*(is-a)* $= Temperature : H_U$*(is-a)*, $Humidity : H_1$*(is-a)*$\neq Wind : H_U$*(is-a)*, $Rainy : H_1$*(is-a)* $\not\leq LightRain : H_U$*(is-a)*, $HeavyRain : H_2$*(is-a)*$\leq Rain : H_U$*(is-a)*, etc.

Definition: A *user perspective* UP with respect to a set of ontologies $O_1, \cdots, O_K$ is defined by a user ontology O_U and a set of interoperation constraints IC from hierarchies in $O_1, \cdots, O_K$ to hierarchies in user ontology O_U.

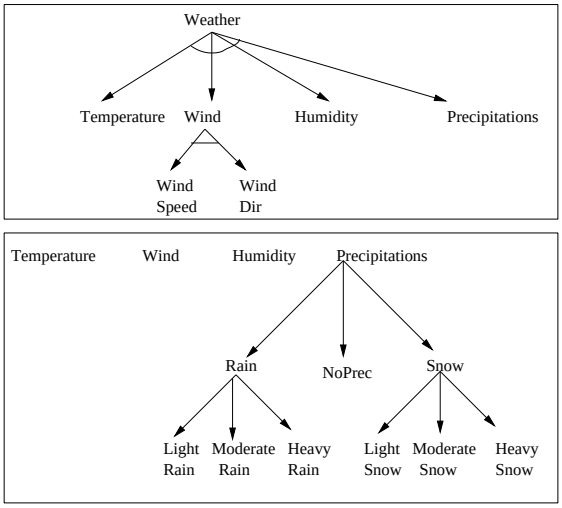

Fig. 5. User ontology O_U

We write $UP = (O_U, IC)$. In particular, the ontologies $O_1, \cdots, O_K$ and O_U could be simply hierarchies.

Definition: Let $(H_1, \preceq_1), \cdots, (H_K, \preceq_K)$ be a set of K hierarchies and $UP = (H_U, IC)$ a user perspective with respect to the hierarchies $H_1, \cdots, H_K$. We say that the hierarchies $H_1, \cdots, H_K$ are *integrable* according to the hierarchy $(H_U, \preceq)$ in the presence of the interoperation constraints IC (or equivalently H_U is the *integration hierarchy* of $H_1, \cdots, H_K$) if there exist K injective partial mappings $\phi_1, \cdots, \phi_K$ from $H_1, \cdots, H_K$, respectively, to H_U with the following two properties:

- For all $x, y \in H_i$, if $x \preceq_i y$ then $\phi_i(x) \preceq \phi_i(y)$ (we call this *order preservation*);
- For all $x \in H_i$ and $y \in H_U$, if $(x : H_i \text{ op } y : H_U) \in IC$, then $\phi_i(x)$ op y in the hierarchy H_U (we call this *interoperation constraints preservation*).

Thus, a set of ontologies are integrable from a user perspective, if a set of mappings from the hierarchies in the local ontologies to the user hierarchies in the user ontology (satisfying the properties in the integration hierarchy definition) can be found.

We propose a simple algorithm for finding a set of mappings that witness the integration of the hierarchies $H_1, \cdots, H_K$ according to a user perspective $UP = (O_U, IC)$ (see Figure 6). It is easy to check if the set of mappings found by this algorithm is consistent with the interoperation constraints and if it satisfies the order preservation property (see [15] for the details of the algorithm). We use the resulting set of mappings to integrate a set of ontologies $O_1, \cdots, O_K$ according to a user ontology O_U in the presence of the interoperation constraints $IC = \{IC(ord) | ord \in \Lambda\}$.

Example: Let H_1, H_2 and H_U be the *is-a* hierarchies in Figures 3, 4 and 5, respectively. Let *IC(is-a)*= $\{Temp : H_2\textit{(is-a)} = Temperature : H_U\textit{(is-a)},$

Finding Mappings
Input: a set of hierarchies $H_1, \cdots, H_K$ and a user perspective $UP = (H_U, IC)$.
Output: a mappings set MS.
{
$MS = \phi$
for (each H_i)

Name Matching Mappings:
for (each $term \in H_i$)
If ($term \in H_U$),then
$MS \rightarrow MS \cup \{term : H_i \rightarrow term : H_U\}$
(unless there is a constraint that does not allow this)

Equality Constraints Mappings:
for (each equality constraint $term_1 : H_i = term_2 : H_U$)
$MS \rightarrow MS \cup \{term_1 : H_i \rightarrow term_2 : H_U\}$
If (MS is consistent with the non-equality constraints)
return MS
Else
eliminate mappings that are inconsistent with the integrity constraints
return MS
}

Fig. 6. Algorithm for finding mappings between a set of data source hierarchies and a user hierarchy

$Outlook : H_1(\text{is-a}) = Precipitations : H_U(\text{is-a})$, $Prec : H_2(\text{is-a}) = Precipitations : H_U(\text{is-a})$, $Sunny : H_1(\text{is-a}) = NoPrec : H_U(\text{is-a})$, $Rainy : H_1(\text{is-a}) \not\leq Rain : H_U(\text{is-a})$, $LightRain : H_2(\text{is-a}) \leq Rain : H_U(\text{is-a})$, $Rainy : H_1(\text{is-a}) \not\leq Rain : H_U(\text{is-a}), \cdots\}$. According to the first step of the Finding Mappings algorithm (name matching mappings), we add the mappings in Table 3. According to the second step of the algorithm (equality constraint mappings), we add the mappings in Table 4. We can easily check that all the mappings constructed are consistent with the non-equality constraints and satisfy the order preservation property.

Table 3. Mappings from $H_1(\text{is-a})$ and $H_2(\text{is-a})$ (corresponding to the data sets D_1 and D_2, respectively) to $H_U(\text{is-a})$ found using name matching strategy.

ϕ_1	ϕ_2
$Temperature \rightarrow Temperature$	-
$Wind \rightarrow Wind$	$Wind \rightarrow Wind$
$Humidity \rightarrow Humidity$	$Humidity \rightarrow Humidity$
-	$Rain \rightarrow Rain$
-	$LightRain \rightarrow LightRain$
-	$ModerateRain \rightarrow ModerateRain$
-	$HeavyRain \rightarrow HeavyRain$
-	$LightSnow \rightarrow LightSnow$
-	$ModerateSnow \rightarrow ModerateSnow$
-	$HeavySnow \rightarrow HeavySnow$
-	$NoPrec \rightarrow NoPrec$

Once a set of mappings is found using the algorithm in Figure 6, the user is given the opportunity to inspect the mappings and add other mappings if needed and if they don't violate the interoperation constraints or the order preservation property.

Table 4. Mappings from H_1*(is-a)* and H_2*(is-a)* (corresponding to the data sets D_1 and D_2, respectively) to H_U*(is-a)* found from equality constraints.

ϕ_1	ϕ_2
-	$Temp \rightarrow Temperature$
$Outlook \rightarrow Precipitations$	$Prec \rightarrow Precipitations$
$Sunny \rightarrow NoPrec$	-
$Rainy \rightarrow Rain$	-

3.3 Conversion Functions

So far, we have defined ontologies, explained what it means to integrate ontologies and showed how a user can check if his or her ontology can be an integration for a set of ontologies associated with autonomous data sources. Once the user integration ontology is defined (together with the mapping to the data sources ontologies), the user's goal is to ask queries in his/her ontology and get sound answers from the data sources. For example, in the weather example, the user may want to ask queries about the days when the *Temperature* was higher than 40F. To get the answer to such a query, besides name mappings ($Temp : O_2 \rightarrow Temperature : O$), a conversion from degree Celsius to Fahrenheit is needed in the case of the second data source D_2.

Definition [14, 15]: We define $T = \{\tau \mid \tau \text{ is a string}\}$ to be a set of *types*. For each type τ, $dom(\tau) = \{\text{v} \mid \text{v is a value of type } \tau\ \}$ is called the *domain* of τ. The members of *dom(*τ*)* are called *values* of type τ. For example, type τ could be a predefined type, e.g., *int* or *string* or it can be a type like F^o (degrees Fahrenheit), USD (US dollars), *mph* (miles per hour) or it can be an enumerated type such as *Outlook* whose domain is given by the values: *Sunny, Rainy, Snowy* etc.

Definition: We say that a total function $\tau_1 2\tau_2 : dom(\tau_1) \rightarrow dom(\tau_2)$ that maps values of τ_1 to values of τ_2 is a *conversion function* from τ_1 to τ_2. The set of all conversion functions must satisfy the following constraints:

- For every two types τ_i, $\tau_j \in \mathcal{T}$ at most one conversion function $\tau_i 2\tau_j$ exists.
- For every type $\tau \in \mathcal{T}$, $\tau 2\tau$ exists (the identity function).
- If $\tau_i 2\tau_j$ and $\tau_j 2\tau_k$ exist, then $\tau_i 2\tau_k$ exists and $\tau_i 2\tau_k = \tau_i 2\tau_j \circ \tau_j 2\tau_k$.

We say that τ_1 can be *converted* into τ_2 and we write $\tau_1 \rightarrow \tau_2$ if there exists a conversion function $\tau_1 2\tau_2$. Note that, if τ_1 and τ_2 are on the same path in a hierarchy $(H, \leq)$ and $\tau_1 \leq \tau_2$, then $\tau_1 \rightarrow \tau_2$, which means that $\tau_1 2\tau_2$ exists (it could be the identity.) A user needs to specify conversion functions for all the ontology mappings defined in the system. If a conversion function is not explicitly specified, it is assumed to be the identity function.

Example: The conversion function associated with the mapping *Humidity*:$O_1 \rightarrow$ *Humidity*:O_U is the identity. The conversion function associated with the mapping *Temp*:$O_2 \rightarrow$*Temperature*:O_U (where *Temp* is measured in degrees Celsius and *Temperature* is measured in degrees Fahrenheit) is the function *Temp(C)2Temperature(F)* which converts *Celsius* to *Fahrenheit*.

Definition: Let H be a hierarchy and τ a type in that hierarchy. We define $\text{below}_H(\tau)$ as being the union between the values of τ and the subtypes τ' of τ, i.e., $\text{below}_H(\tau) := \{\tau' | \tau' \in H, \tau' \leq_H \tau\} \cup \text{dom}(\tau)$. If $\tau' \in \text{below}_H(\tau)$, we say that τ implies a higher level of abstraction than τ' or, equivalentely, τ' implies a lower level of abstraction than τ. The level of abstraction at which instances in a data source are specified determines a *cut* through the associated data-source ontology.

Example: We have $below_H(Prec)$={*Rain, NoPrec, Snow, LightRain, ModerateRain, HeavyRain, LightSnow, ModerateSnow, HeavySnow*}. Furthermore, *Rain* implies a higher level of abstraction than *LightRain*, as *LightRain* is below *Rain* in the hierarchy associated with the attribute *Precipitation* in the ontology O_2 corresponding to the data source D_2 in the weather example. The set $\{Rain, NoPrec, Snow\}$ represents a cut through the hierarchy associated with the attribute *Precipitation* in the same ontology.

Definition: Let τ_1 and τ_2 be two types. A type τ is called the *least common supertype* of τ_1 and τ_2 if:

- $\tau_1 \rightarrow \tau$ and $\tau_2 \rightarrow \tau$.
- If there exists τ' such that $\tau_1 \rightarrow \tau'$ and $\tau_2 \rightarrow \tau'$, then $\tau \rightarrow \tau'$.

Example: Let $X = Rain$ and $Y = HeavySnow$ be two terms in the *is-a* hierarchy of the user ontology in the *Weather* example. Then the least common supertype of $type(X)$ and $type(Y)$ is *Precipitation*.

3.4 Ontology-Extended Data Sources

We will show that we can ensure the semantical correctness of an answer to a query if we extend each data source with its corresponding ontology and also with the type information associated with each attribute (i.e., data source schema), and specify conversion functions between different types.

Definition: Let $\{A_1, \cdots, A_n\}$ be the set of attributes used to describe the data in a particular data source D, and let $\{\tau_1, \cdots, \tau_n\}$ be the set of types associated with these attributes. The set $\{A_1 : \tau_1, \cdots, A_n : \tau_n\}$ is called the *schema* of the data source D.

Definition: Two schemas S_1=(A_1:τ_1^1, $\cdots$, A_n:τ_n^1) and S_2=(A_1:τ_1^2, $\cdots$, A_n:τ_n^2) are *compatible* if τ_i^1 and τ_i^2 have a *least common supertype* τ_i and the conversion functions $\tau_i^1 2\tau_i$ and $\tau_i^2 2\tau_i$ exist for all $i = 1, \cdots, n$. The common schema $S = (A_1 : \tau_1, \cdots, A_n : \tau_n)$ is called the *least common super-schema* of S_1 and S_2. The conversion functions $S_j 2S$ are defined by:

$S_j 2S(D) = \{(\tau_1^j 2\tau_1(x_1), \cdots, \tau_n^j 2\tau_n(x_n)) | (x_1, \cdots, x_n) \in D\}$ for $j = 1, 2$.

Definition: We say that *(D,S,O)* is an *ontology-extended data source* if D is a data source (represented as a table), O is an ontology over D, $S = \{A_1 : \tau_1, \cdots, A_n : \tau_n\}$ is the data source schema, and the following conditions are satisfied:

(1) $\tau_1, \cdots, \tau_n \in O$ are types in the ontology O and
(2) $D \subseteq \text{below}_O(\tau_1) \times \cdots \times \text{below}_O(\tau_n)$.

3.5 Statistical Query Language

So far, we have extended data sources with ontologies and type information. We want to use these ontology-extended data sources to answer statistical queries.

Definition: We define a *statistical query language* consisting of a set of traditional *data operators* and a set of *statistical operators* that are used to formulate statistical queries. The set of data operators consists of set operators (e.g., $UNION$, $INTERSECTION$, etc.) and relational operators (e.g., $SELECT$, $PROJECT$, etc.) that are used to specify the data to which the statistical operators are applied. The set of statistical operators consists of aggregate operators (e.g., AVG, $COUNT$, MIN, MAX), used to compute aggregate statistics for a data set and compositional operators (e.g., $+$, $UNION$, etc.), used to combine statistics collected from several data sources.

To ensure that the answers to statistical queries are sound, we need to make sure that the results of the operators defined above are well-typed. Bonatti and his collegues [14] showed how one can ensure that the results of data operators are well-typed. In short, the result of a unary operator is always well-typed. The result of a binary data operator is well-typed if the data sources to which the operator is applied have a least common super-schema. The results of statistical operators are well-typed if the data sources to which they are applied are well-typed and their schemas have a least common super-schema.

3.6 An Example Demonstrating Statistical Queries over Ontology-Extended Data Sources

In this section we will show how we can answer statistical queries needed to construct Naive Bayes classifiers from semantically heterogeneous data. Assume there exist two data sources D_1 and D_2 with the associated ontologies O_1 and O_2 and a user is interested in analyzing the data from D_1 and D_2 from his perspective, which corresponds to the ontology O_U and a set of interoperation constraints IC. Suppose D_1 contains 10 instances of *Rainy* days and 30 instances of *Snowy* days. The data source D_2 contains 10 instances of *LightRain* days, 20 instances of *HeavyRain* days, 10 instances of *LightSnow* days and 10 instances of *HeavySnow* days.

A statistical query q^{O_U} is posed to the two data sources based on the ontology O_U: What fraction of the days are *Rain* days? After performing the necessary mappings ($Rainy : O_1 \rightarrow Rain : O_U$, $Rain : O_2 \rightarrow Rain : O_U$), the answer to this query can be computed in a straightforward way as the ratio of the number of *Rain* days (20+10+20=50) divided by the total number of days (100) yielding an answer of 0.5.

Now consider another query r^{O_U} (also based on the ontology O_U): What fraction of days are *HeavyRain* days? The answer to this query is not as straightforward as the answer to the previous query q_{O_U}. This is due to the fact that the quantification of rain for the days in data source D_1 is only *partially specified* [16] with respect to the ontology O_U. Consequently, we can never know the precise fraction of days that are *HeavyRain* days based on the information

available in the two data sources. However, if it is reasonable to assume that the data contained in both D_1 and D_2 are drawn from the same universe (i.e., can be modeled by the same underlying distribution), we can estimate the fraction of days that are *HeavyRain* days in the data source D_1 based on the fraction of *Rain* days that are *HeavyRain* days in the data source D_2 (i.e., 20 out of 30) and use the result to answer the query r^{O_U}. Under the assumption that the samples of days in D_1 and D_2 can be modeled by the same distribution, the estimated number of *HeavyRain* days in D_1 is given by $\left(\frac{20}{30}\right)(20) = \left(\frac{40}{3}\right)$. Hence, the estimated number of *HeavyRain* days in D_1 and D_2 is $\left(\frac{40}{3}\right) + 20 = \left(\frac{100}{3}\right)$. Thus, the answer to the query r^{O_U} is $\left(\frac{100}{3}\right)\left(\frac{1}{100}\right) = \frac{1}{3}$. While the assumption that the data sources under consideration can be modeled by the same underlying distribution may be reasonable in some cases, in other cases, alternative assumptions may be justified. For example, some users might want to assume that the precise amount of rain in data source D_1 cannot reasonably be estimated on the basis of the rain distribution of the days in data source D_2 and hence require that the answer to query r^{O_U} be based only on the data in D_2, yielding an answer of 20 out of 100 or 0.2.

Note that the answer to query q^{O_U} is completely determined by the ontologies O_1, O_2, O_U, the mappings shown in Tables 3, 4 and the data available in the data sources D_1 and D_2. However, answer to the query r^{O_U} is only partially determined by the ontologies O_1, O_2, O_U, the mappings shown in Tables 3, 4 and the data available in the data sources D_1 and D_2. In such cases, answering statistical queries from semantically heterogeneous data sources requires the user to supply not only the mappings between ontologies associated with the data sources and his or her ontology, but also additional assumptions of a statistical nature (e.g., that data in D_1 and D_2 can be modeled by the same underlying distribution). The validity of the answer returned depends on the validity of the assumptions and the soundness of the procedure that computes the answer based on the supplied assumptions.

Let $(D_1, S_1, O_1), \cdots, (D_K, S_K, O_K)$ be K ontology-extended data sources and O_U a user ontology. Let $Z(O_1), \cdots, Z(O_K)$ be the levels of abstraction (cuts) at which the instances are specified in the data sources $D_1, \cdots, D_K$, respectively and $Z(O_U)$ a cut through the user ontology defining the level of abstraction at which the user queries are formulated. When answering statistical queries from $D_1, \cdots, D_K$ using the user ontology O_U, the name and type heterogeneity problems are solved once valid mappings between data source ontologies and user ontology have been specified. However, we still encounter problems as those described in the above. More precisely, having different ontologies associated with different data sources implies that the instances could be specified at different levels of abstraction with respect to a user ontology.

Definition: Let $x = (v_{A_1}, \cdots, v_{A_n}) \in D_j$ be an instance in D_j. We say that the instance x is:

- *completely specified* if for all $1 \leq i \leq n$, the correspondent of v_{A_i} in O_U belongs to the user level of abstraction $Z(O_U)$.

- *partially specified* if there exist at least one attribute value v_{A_i} for which the corresponding value in $Z(O_U)$ does not belong to the user level of abstraction $Z(O_U)$. This value can be *under-specified* if its correspondent in the user ontology is above the user cut, or *over-specified* if its correspondent in the user ontology is below the user cut (but it actually does not exist).

Example: Assume that the instances in the data source D_1 are specified in terms of $Rain$, $NoPrec$ and $Snow$. The instances in D_2 are specified in terms of $LightRain$, $ModerateRain$, $HeavyRain$, $NoPrec$, $LightSnow$, $ModerateSnow$, $HeavySnow$. Assume that according to the user level of abstraction the instances have to be specified in terms of $LightRain$, $ModerateRain$, $HeavyRain$, $NoPrec$ and $Snow$. We can see that in this case, the instances in D_1 are under-specified, while the instances in D_2 are over-specified. Thus, $Rain$ is an under-specified value of the attribute $Prec$ in D_1, while $LightSnow$, $ModerateSnow$, $HeavySnow$ are over-specified values of the attribute $Prec$ in D_2.

One way to deal with the under- or over-specification problems is to replace the original data set with a new data set, where the values of the attributes are at the right level of specification, given the user level of abstraction. In principle, this can be easily done when an attribute is over-specified: we replace the over-specified value with a higher level ancestor in the corresponding hierarchy (specifically, with the ancestor that has the same level of abstraction as the value in the user hierarchy). However, for the under-specified values, additional assumptions need to be made by the user (e.g., all data comes from the same distribution) and under-specified values are filled accordingly, by replacing the original instance with a new instance having the right level of specification, according to a distribution corresponding to the user preference. This way of handling partially specified data, together with the mappings and conversion functions ensure correct answers to statistical queries posed over distributed, semantically heterogeneous data sources.

Now we show how Naive Bayes classifiers can be generated from semantically heterogeneous, horizontally distributed data. Let $A_1(O_U), \cdots, A_n(O_U)$ be the user attributes with respect to a data domain and $O_U = \{H_1(A_1), \cdots, H_n(A_n)\}$ the user ontology associated with these attributes. Let $v_{A_1}(O_U), \cdots, v_{A_n}(O_U)$ be a learning cut through the user ontology (note that $v_{A_i}(O_U) \subseteq H_U(A_i)$ could be a set of values of the attribute $A_i(O_U)$). If the data is horizontally distributed, then each data source D_j contains an attribute $A_i(O_j)$ that maps to $A_i(O_U)$.

The algorithm for learning naive Bayes classifiers from horizontally distributed heterogeneous data sources is similar to the algorithm for learning naive Bayes classifiers form horizontally distributed homogeneous data sources [4, 15]. As opposed to this scenario, in the case of heterogeneous data sources: First, the set of mappings is used to find the correspondents of the user attributes in the distributed data sources (e.g., $A_i(O_j) \rightarrow A_i(O_U)$) and also to resolve the semantic mismatches between the correspondent attributes. Second, for each attribute value $v \in v_{A_i}(O_U)$ in the user cut, we compute the counts at a particular data source D_j that contains that attribute, as follows:

– If v is over-specified in D_j, then we recursively propagate up the counts from its children in $H_i(D_j)$ to v, till all the children are specified in D_j (primitives). For example, in Figure 4, to compute the counts in D_2 corresponding to $Snow$, we compute the counts for $LightSnow$, $ModerateSnow$, and $HeavySnow$ and we add them up.
– If v is under-specified in D_j, we can treat it as a missing value and thus we reduce our problem to the problem of filling in missing values. Under the assumption that all the data is coming from the same distribution, we can estimate this distribution based on a data set where the values are specified, and then propagate down the counts based on that distribution in a data set where the values are under-specified. For example, if there are 8 instances in D_1 for which $Prec$ takes value $Rain$ and if the distribution over the values $LightRain$, $ModerateRain$, $HeavyRain$ is $(25, 50, 25)$, then we can infer that there are 2 instances for which $Prec = LightRain$, 4 instances for which $Prec = ModerateRain$ and 2 instances for which $Prec = HeavyRain$.

Once the counts are estimated this way, the algorithm works as in the case of homogeneous distributed data. Thus, we can see that we don't need to explicitly construct data sets where all the instances are completely specified, as the counts can be computed implicitly.

4 Summary and Discussion

In this paper, we showed how the approach for learning from distributed data sources introduced in [4] can be extended to yield an approach for learning from heterogeneous data sources, by presenting a way to answer statistical queries needed by learning algorithms from heterogeneous data. To do that, we defined ontologies, user perspective and integration of a set of ontologies from a user perspective. We associated an ontology with each data source. In this setting, answering statistical queries from ontology-extended data sources implies solving a variant of the information integration problem [2] together with a way of handling partially specified data that appears when different data sources are specified at different levels of abstraction [16]. We defined a statistical query language and ensured that the invocation of the operators in this language results in well-typed data sets or statistics over data sets, through the means of mappings and conversion functions between terms in different ontologies. We demonstrated our approach by designing an algorithm for generating Naive Bayes classifiers from distributed, semantically heterogeneous data.

In terms of related work, Davidson *et al.* [17] and Eckman [18] survey alternative approaches to data integration. Most of the traditional information integration approaches use mediator programs to integrate heterogeneus data sources. However, these approaches are not theoretically well founded. Levy [2] proposed an approach based on logic, which is theoretically well-founded, but it doesn't deal with type heterogeneity.

Our definition of ontology-extended data sources was inspired by a similar definition for ontology-extended relational algebra introduced in [14]. The authors in [14] associate a graph with each hierarchy. In their setting, the user defines a set of mappings between different hierarchies in the system and a set of interoperation constraints. The mappings are used to merge all the individual graph hierarchies into an overall graph hierarchy. An integration hierarchy is given by a canonical hierarchy which consists of all strongly connected components in the graph hierarchy. An integration hierarchy is valid if it satisfies a set of interoperation constraints and order preservation property.

As opposed to [14], we define a user perspective as consisting of a user ontology and a set of interoperation constraints. We present a simple algorithm for coming up with mappings between data source ontologies and a user ontology based on interoperation constraints and an algorithm for checking that these mappings are valid. Our approach is more general that the approach in [14] because users can impose their own perspective over a set of data sources, which ensures flexibility required for Semantic Web applications where different users may want to access data from different perspectives or for that matter, even the same user may impose different ontologies in different contexts.
McClean *et al.* [19, 20] provides an approach to answering aggregate queries formulated in a user ontology, from statistical databases. Their results are similar to our results. However, their framework assumes that there exists metadata, in terms of mappings between ontologies, in the system, while we give the user the possibility to specify how he or she wants to use the existent data, by specifying a set of interoperation constraints that relates data of interest. Another strength of our approach comes from the ability to deal with type heterogeneity (by using conversion functions, e.g. $F \rightarrow C$).

Our approach to learning from ontology-extended data sources is similar to the approach in [16], where AVT's are associated with the attributes in a data set and the level of abstraction which gives the best accuracy is sought. In our case, we assume the level the abstraction is given by the user. This level defines a level of abstraction for each data source ontology, which results in some attributes being over-specified while others might be under-specified, hence the connection with learning from partially specified data. We can envision scenarios where there is no user predefined level of abstraction, in which case we would iterate through successive user levels of abstraction as in [16] and the one that gives the best accuracy is chosen.

Directions for future work include the extension of the approach presented in this paper to other types of ontologies besides attribute values taxonomies and applications to problems in bioinformatics.

Acknowledgments. This work has been supported in part by grants from the National Science Foundation (IIS 0219699) and the National Institutes of Health (GM 066387) to Vasant Honavar.

References

1. Hendler, J.: Science and the semantic web. Science **299** (2003)

2. Levy, A.Y.: Logic-based techniques in data integration. In: Logic-based artificial intelligence. Kluwer Academic Publishers (2000) 575–595
3. Reinoso-Castillo, J., Silvescu, A., Caragea, D., Pathak, J., Honavar, V.: Information extraction and integration from heterogeneous, distributed, autonomous information sources: A federated, query-centric approach. In: IEEE International Conference on Information Integration and Reuse, In press (2003)
4. Caragea, D., Silvescu, A., Honavar, V.: 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)
5. Casella, G., Berger, R.: Statistical Inference. Duxbury Press, Belmont, CA (2001)
6. Mitchell, T.: Machine Learning. McGraw Hill (1997)
7. Pearl, J.: Graphical Models for Probabilistic and Causal Reasoning. Cambridge Press (2000)
8. Jensen, F.V.: Bayesian Networks and Decision Graphs. Springer (2001)
9. Quinlan, R.: Induction of decision trees. Machine Learning **1** (1986) 81–106
10. Friedman, N., Getoor, L., Koller, D., Pfeffer, A.: Learning probabilistic relational models. In: Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, Morgan Kaufmann Publishers Inc. (1999) 1300–1309
11. Atramentov, A., Leiva, H., Honavar, V.: Learning decision trees from multi-relational data. In Horvth, T., Yamamoto, A., eds.: Proceedings of the 13th International Conference on Inductive Logic Programming. Volume 2835 of Lecture Notes in Artificial Intelligence., Springer-Verlag (2003) 38–56
12. Silvescu, A., Andorf, C., Dobbs, D., Honavar, V.: Inter-element dependency models for sequence classification. In: ICDM, Submitted (2004)
13. Agrawal, R., Shafer, J.C.: Parallel Mining of Association Rules. IEEE Transactions On Knowledge And Data Engineering **8** (1996) 962–969
14. Bonatti, P., Deng, Y., Subrahmanian, V.: An ontology-extended relational algebra. In: Proceedings of the IEEE Conference on INformation Integration and Reuse, IEEE Press (2003) 192–199
15. Caragea, D.: Learning from Distributed, Heterogeneous and Autonomous Data Sources. PhD thesis, Department of Computer Sciene, Iowa State University, USA (2004)
16. Zhang, J., Honavar, V.: Learning naive bayes classifiers from attribute-value taxonomies and partially specified data. In: Proceedings of the Conference on Intelligent System Design and Applications, In Press (2004)
17. Davidson, S., Crabtree, J., Brunk, B., Schug, J., Tannen, V., Overton, G., Stoeckert, C.: K2/kleisli and gus: Experiments in integrated access to genomic data sources. IBM Journal **40** (2001)
18. Eckman, B.: A practitioner's guide to data management and data integration in bioinformatics. Bioinformatics (2003) 3–74
19. McClean, S., Páircéir, R., Scotney, B., Greer, K.: A Negotiation Agent for Distributed Heterogeneous Statistical Databases. SSDBM 2002 (2002) 207–216
20. McClean, S., Scotney, B., Greer, K.: A Scalable Approach to Integrating Heterogeneous Aggregate Views of Distributed Databases. IEEE Transactions on Knowledge and Data Engineering (TKDE) (2003) 232–235