Information Integration and Knowledge Acquisition from Semantically Heterogeneous Biological Data Sources

Doina Caragea[1,4], Jyotishman Pathak[1,4], Jie Bao[1,4], Adrian Silvescu[1,4], Carson Andorf[1,3,4], Drena Dobbs[2,3,4], and Vasant Honavar[1,2,3,4]

[1] AI Research Laboratory, Department of Computer Science, 226 Atanasoff Hall
[2] Department of Genetics, Development and Cell Biology, 1210 Molecular Biology
[3] Bioinformatics and Computational Biology Program, 2014 Molecular Biology
[4] Computational Intelligence, Learning and Discovery Program, 214 Atanasoff Hall
Iowa State University, Ames, IA 50011
honavar@cs.iastate.edu

Abstract. We present INDUS (Intelligent Data Understanding System), a federated, query-centric system for knowledge acquisition from autonomous, distributed, semantically heterogeneous data sources that can be viewed (conceptually) as tables. INDUS employs ontologies and inter-ontology mappings, to enable a user or an application to view a collection of such data sources (regardless of location, internal structure and query interfaces) as though they were a collection of tables structured according to an ontology supplied by the user. This allows INDUS to answer user queries against distributed, semantically heterogeneous data sources without the need for a centralized data warehouse or a common global ontology. We used INDUS framework to design algorithms for learning probabilistic models (e.g., Naive Bayes models) for predicting GO functional classification of a protein based on training sequences that are distributed among SWISSPROT and MIPS data sources. Mappings such as EC2GO and MIPS2GO were used to resolve the semantic differences between these data sources when answering queries posed by the learning algorithms. Our results show that INDUS can be successfully used for integrative analysis of data from multiple sources needed for collaborative discovery in computational biology.

1 Introduction

Ongoing transformation of biology from a data-poor science into an increasingly data-rich science has resulted in a large number of autonomous data sources (e.g., protein sequences, structures, expression patterns, interactions). This has led to unprecedented, and as yet, largely unrealized opportunities for large-scale collaborative discovery in a number of areas: characterization of macromolecular sequence-structure-function relationships, discovery of complex genetic regulatory networks, among others.

Biological data sources developed by autonomous individuals or groups differ with respect to their ontological commitments. These include assumptions

concerning the *objects* that exist in the *world*, the *properties* or *attributes* of the objects, *relationships* between objects, the possible *values* of attributes, and their *intended meaning*, as well as the *granularity* or *level of abstraction* at which objects and their properties are described [17]. Therefore, *semantic differences* among autonomous data sources are simply unavoidable. Effective use of multiple sources of data in a given context requires reconciliation of such semantic differences. This involves solving a data integration problem. Development of sound approaches to solving the information integration problem is a prerequisite for realizing the goals of the Semantic Web as articulated by Berners-Lee et al. [5]: seamless and flexible access, integration and manipulation of semantically heterogeneous, networked data, knowledge and services.

Driven by the semantic Web vision, there have been significant community-wide efforts aimed at the construction of ontologies in life sciences. Examples include the Gene Ontology (www.geneontology.org) [2] in biology and Unified Medical Language System (www.nlm.nih.gov/research/umls) in heath informatics. Data sources that are created for use in one context often find use in other contexts or applications (e.g., in collaborative scientific discovery applications involving data-driven construction of classifiers from semantically disparate data sources [9]). Furthermore, users often need to analyze data in different contexts from different perspectives. Therefore, there is no single privileged ontology that can serve all users, or for that matter, even a single user, in every context. Effective use of multiple sources of data in a given context requires flexible approaches to reconciling such semantic differences from the user's point of view.

Against this background, we have investigated a federated, query-centric approach to information integration and knowledge acquisition from distributed, semantically heterogeneous data sources, from a user's perspective. The choice of the federated, query-centric approach was influenced by the large number and diversity of data repositories involved, together with the user-specific nature of the integration tasks that need to be performed. Our work has led to INDUS, a system for information integration and knowledge acquisition (see Figure 1). INDUS relies on the observation that both the information integration and knowledge acquisition tasks can be reduced to the task of answering queries from distributed, semantically heterogeneous data sources. We associate ontologies with data sources and users and show how to define mappings between them. We exploit the ontologies and the mappings to develop sound methods for flexibly querying (from a user perspective) multiple semantically heterogeneous distributed data sources in a setting where each data source can be viewed (conceptually) as a single table [10, 9].

The rest of the paper is organized as follows: Section 2 introduces the problem we are addressing more precisely through an example. Section 3 describes the design and the architecture of INDUS. Section 4 demonstrates how INDUS can be used for knowledge acquisition tasks using as an example a simple machine learning algorithm (Naive Bayes). We end with conclusions, discussion of related work and directions for future work in Section 5.

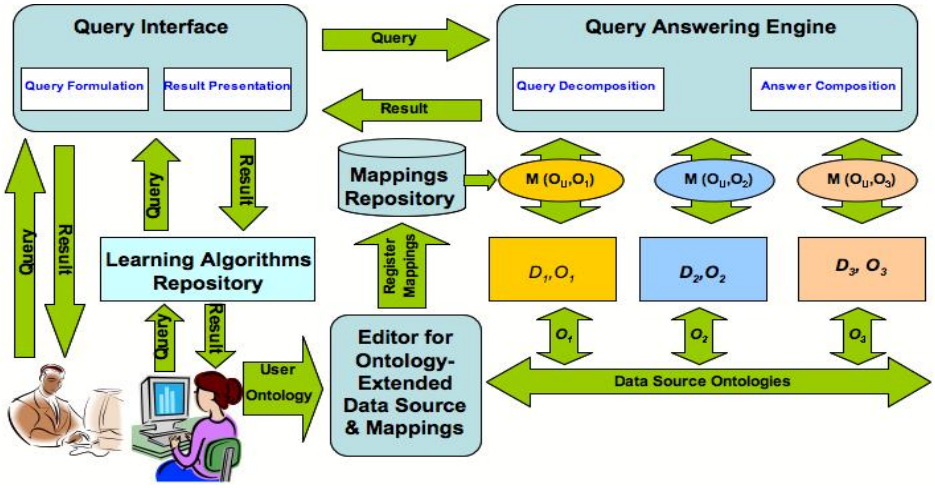

Fig. 1. INDUS: a system for information integration and knowledge acquisition from semantically heterogeneous distributed data. Queries posed by the user are answered by a query answering engine, which uses mappings between the user ontology and the data source ontologies to resolve semantic differences. A user-friendly editor is used to specify ontologies and mappings between ontologies.

2 Illustrative Example

The problem that we are wish to address is best illustrated by an example. Consider several biological laboratories that independently collect information about *Protein Sequences* in connection to their *Structure* and *Function*. Suppose that the data D_1 collected by a first laboratory contains human proteins and it is described by the attributes *Protein ID*, *Protein Name*, *Protein Sequence*, *Prosite Motifs* and *EC Number* (stored as in Table 1). The data D_2 collected by a second laboratory contains yeast proteins and it is described by the attributes *Accession Number AN*, *Gene*, *AA Sequence*, *Length*, *Pfam Domains*, and *MIPS Funcat* (stored as in Table 2). A data set D_3 collected by a third laboratory contains both human and yeast proteins and it is described by the attributes *Entry ID*, *Entry Name*, *Organism*, *CATH Domains* and *CATH Classes* corresponding to the domains (stored as in Table 3).

Consider a biologist (user) U who wants to assemble a data set based on the data sources of interest D_1, D_2, D_3, from his or her own perspective, where the representative attributes are *ID*, *Source*, *AA composition* (a.k.a. amino acid distribution, i.e. number of occurrences of each amino acid in the amino acid sequence corresponding to the protein), *Structural Classes* and *GO Function*. This requires the ability to manipulate the data sources of interest from the user's perspective. However, the three data sources differ in terms of semantics from the user's perspective. In order to cope with this heterogeneity of semantics,

Table 1. Data D_1 containing human proteins collected by a laboratory Lab_1

Protein ID	*Protein Name*	*Protein Sequence*	*Prosite Motifs*	*EC Number*
P35626	Beta-adrenergic receptor kinase 2	MADLEAVLAD VSYLMAMEKS ...	*RGS* *PROT_KIN_DOM* *PH_DOMAIN*	2.7.1.126 Beta-adrenergic receptor kinase
Q12797	Aspartyl/asparaginyl beta-hydroxylase	MAQRKNAKSS GNSSSSGSGS ...	*TPR* *TPR_REGION* *TRP*	1.14.11.16 Peptide-aspartate beta-dioxygenase
Q13219	Pappalysin-1	MRLWSWVLHL GLLSAALGCG ...	*SUSHI*	3.4.24.79 Pappalysin-1
...	...	...	...	...

Table 2. Data D_2 containing yeast proteins collected by a laboratory Lab_2

AN	*Gene*	*AA Sequence*	*Length*	*Pfam Domains*	*MIPS Funcat*
P32589	SSE1	STPFGLDLGN NNSVLAVARN ...	692	*HSP*70	16.01 protein binding
P07278	BCY1	VSSLPKESQA ELQLFQNEIN ...	415	*cNMP_binding* *RIIa*	16.19.01 cyclic nucleotide binding (cAMP, cGMP, etc.)
...	...	...	...	...	...

Table 3. Data D_3 containing human and yeast proteins collected by a laboratory Lab_3

Entry ID	*Entry Name*	*Organism*	*CATH Domains*	*CATH Classes*
P35626	*ARK2_HUMAN*	Human	1omwB0 1omwG0	Mainly beta Few Sec. Struct.
Q12797	*ASPH_HUMAN*	Human	not known	not known
Q13219	*PAPPA_HUMAN*	Human	1jmaB1 1jmaB2	Mainly beta Mainly beta
P32589	*HS78_YEAST*	Yeast	1dkgA1 1dkgA2 1dkgB1	Alpha beta Mainly alpha Alpha beta
P07278	*KAPR_YEAST*	Yeast	1cx4A1 1dkgA2	Alpha beta Alpha beta
...	...	...	...	...

the user must observe that the attributes *Protein ID*, *Accession Number* and *Entry ID*, in the three data sources of interest, are similar to the user attribute *ID*; the attribute *Protein Sequence* in the first data source and the attribute *AA Sequence* in the second data source are also similar and they can be used to infer the user attribute *AA Composition* (by counting the number of occurrences of each amino acid in the corresponding AA sequence); similarly, the attributes *EC Number* and *MIPS Funcat* are similar to the user attribute *GO Function*; finally, the attributes *Organism* and *CATH Classes* in the third data source are similar to the attributes *Source* and *Structural Classes* in the user view.

Therefore, to assemble the user data, one would need to project the data in D_1 (with respect to the attributes *Protein ID*, *Protein Sequence* and *EC Number*) and the data in D_2 (with respect to *AN*, *AA Sequence*, and *MIPS Funcat*) and take the union D_{12} of the resulting sets; then the third data set D_3 needs to be projected with respect to the attributes *Entry Name*, *Organism*, and *CATH Classes*. The cross-product with respect to the common attribute *ID*, between D_{12} and D_3 represents the data that the user is interested in. Notice that all these operations can be written as a query whose result is $D_U = (project(D_1) \cup project(D_2)) \times project(D_3)$. However, before the query can be executed, the semantic differences between values of similar attributes must be resolved.

To establish the correspondence between values that two similar attributes can take, we need to associate types with attributes and map the domain of the type of an attribute to the domain of the type of the corresponding attribute (e.g., *AA Sequence* to *AA Composition* or *EC Number* to *GO Function*). We assume that the type of an attribute can be a standard type such as a collection of values (e.g., amino acids, Prosite motifs, etc.), or it can be given by a simple hierarchical ontology (e.g., species taxonomy). Figure 2 shows examples of (simplified) attribute value hierarchies for the attributes *EC Numbers*, *MIPS Funcat*, and *GO Function* in the data sources D_1, D_2 and the user perspective.

Examples of semantic correspondences in this case could be: *EC 2.7.1.126* in D_1 is equivalent to *GO 0047696* in D_U, *MIPS 16.01* in D_2 is equivalent to *GO 0005515* in D_U and *MIPS 16.19.01* is equivalent to GO *0016208* in D_U. On the other hand, *EC 2.7.1.126* in D_1 is lower than (i.e., hierarchically below) *GO 0004672* in D_U, or for that matter *EC 2.7.1.126* is higher than *GO0004672*. Similarly, *MIPS 16.19.01* in D_2 is lower than *GO 0017076* in D_U, and so on. Therefore the integrated user data D_U could look like in Table 4, where the semantic correspondences have been applied.

In general, the user may want to answer queries such as *the number of human proteins that are involved in kinase activity* from the integrated data or even to infer models based on the data available in order to use them to predict useful information about new unlabeled data (e.g., protein function for unlabeled proteins). INDUS, the system that we have developed in our lab, can be used to answer such queries against distributed, semantically heterogeneous data sources without the need for a centralized data warehouse or a common global ontology. We will describe INDUS in more detail in the next section.

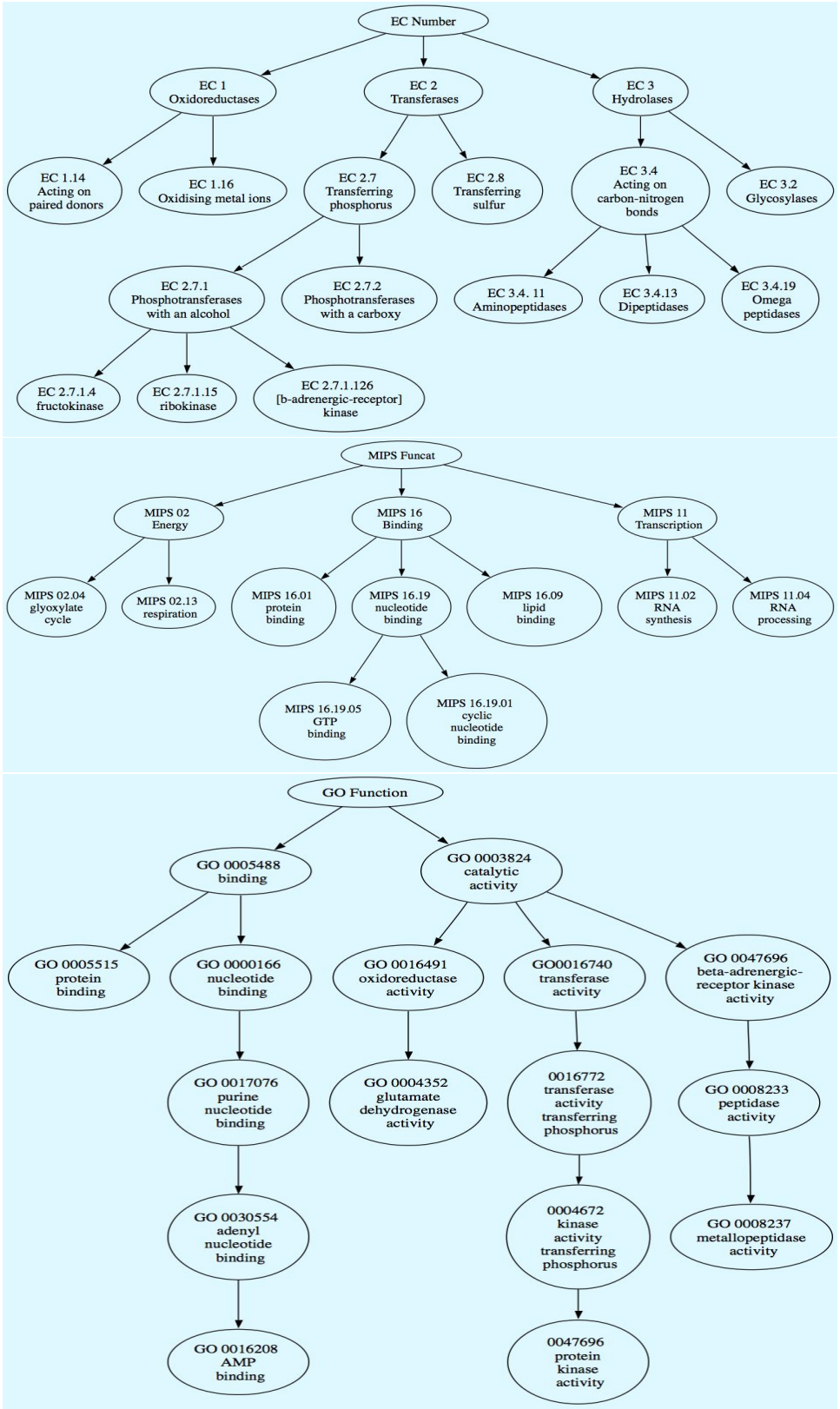

Fig. 2. Ontologies associated with the attributes *EC Number*, *MIPS Funcat* and *GO Function* that appear in the data sources of interest D_1, D_2 and D_U.

Table 4. Integrated user data D_U

ID	*Source*	*AA composition*	*Struct. Classes*	*GO Funct. Class*
P35626	Human	7 3 9 14 ···	Mainly beta Few Sec. Struct.	0047696:beta-adrenergic-receptor kinase activity
Q12797	Human	5 1 7 12 ···	not known	0004597: peptide-aspartate beta-dioxygenase activity
Q13219	Human	10 8 6 15 ···	Mainly beta Mainly beta	0008237: metallopeptidase activity
P39708	Yeast	13 17 18 11 ···	Alpha beta Alpha beta Mainly alpha	0005515: protein binding
Q01574	Yeast	23 16 8 1 ···	Mainly alpha Mainly alpha	0016208: AMP binding
···	···	···	···	···

3 INDUS Design and Architecture

A simplified version of INDUS architecture is shown in Figure 1. As can be seen, several related distributed and semantically heterogeneous *data sources* (servers) can be available to *users* (clients) who may want to query the data sources through a *query interface*. Each user has his or her own view of the domain of interest reflected by a *user ontology*. The system provides default user ontologies (e.g., *GO Function*) and mappings from the data source ontologies to the user ontology (e.g., from *AA Sequence* to *AA Composition* or from *EC Number* to *GO Function*) in a *mapping repository*. However, a user-friendly *ontology and mapping editor* is also available for users if they need to design or modify their own ontologies or mappings (for example, if they need to explore different mappings such as *AA Sequence* to *AA composition* or *AA sequence* to *hydrophobic* versus *hydrophilic AA Composition*).

Once a query is posed by the user, it is sent to a *query answering engine* which acts as a middleware between clients and servers. The query answering engine has access to the data sources in the system and also to the set of mappings available. Thus, when the query answering engine receives a user query, it decomposes this query according to the distributed data sources, maps the individual queries to the data source ontologies, then it composes the results to sub-queries into a final result that is sent back to the user.

The main features of INDUS include:

(1) A clear distinction between data and the semantics of the data: this makes it easy to define mappings from data source ontologies to user ontologies.
(2) User-specified ontologies: each user can specify his or her ontology and mappings from data source ontologies to the user ontology; there is no single global ontology.

(3) A user-friendly ontology and mappings editor: this can be easily used to specify ontologies and mappings; however, a predefined set of ontologies and mappings are also available in a repository.
(4) Knowledge acquisition capabilities: if the information requirements of an algorithm for knowledge acquisition from data (e.g., learning algorithm) can be formulated as statistical queries [10], then such an algorithm can be easily linked to INDUS, making it an appropriate tool for information integration as well as knowledge acquisition tasks.

Some of these features are shared by other systems developed independently, e.g., BioMediator [25]. In the remaining of this section we describe the first three features into more detail, while in the next section we show how INDUS can be used to infer Naive Bayes models.

3.1 Ontology Extended Data Sources

Suppose that the data of interest are distributed over the data sources $D_1, \cdots, D_p$, where each data source D_i contains only a fragment of the whole data D.

Let D_i be a distributed data set described by the set of attributes $\{A_1^i, \cdots, A_n^i\}$ and $O_i = \{\Lambda_1^i, \cdots, \Lambda_n^i\}$ an ontology associated with this data set. The element $\Lambda_j^i \in O_i$ corresponds to the attribute A_j^i and defines the type of that particular attribute. The type of an attribute can be a standard type (e.g., types such as Integer or String; the enumeration of a set of values such as Prosite motifs; etc.) or a hierarchical type, which is defined as an ordering of a set of terms (e.g., the values of the attribute EC number) [6]. Of special interest to us are *isa* hierarchies over the values of the attributes that describe a data source, also called *attribute value taxonomies* (see Figure 2).

The schema S_i of a data source D_i is given by the set of attributes $\{A_1^i, \cdots, A_n^i\}$ used to describe the data together with their respective types $\{\Lambda_1^i, \cdots, \Lambda_n^i\}$ defined by the ontology O_i, i.e., $S = \{A_1 : \Lambda_1, \cdots, A_n : \Lambda_n\}$. We define an *ontology-extended data source* as a tuple $\mathcal{D}_i = <D_i, S_i, O_i>$, where D_i is the actual data in the data source, S_i is the schema of the data source and O_i is the ontology associated with the data source. In addition, the following condition needs also to be satisfied: $D_i \subseteq \Lambda_1^i \times \cdots \times \Lambda_n^i$, which means that each attribute A_j^i can take values in the set Λ_j^i defined by the ontology O_i.

3.2 User Perspective

Let $<D_1,S_1,O_1>, \cdots, <D_p, S_p, O_p>$ be an ordered set of p ontology-extended data sources and U a user that poses queries against the heterogeneous data sources $D_1, \cdots, D_p$. A user perspective is given by a user ontology O_U and a set of semantic correspondences SC between terms in $O_1, \cdots, O_p$, respectively, and terms in O_U. The semantic correspondences can be at attribute level (or schema level), e.g., $A_j^i : O_i \equiv A_l^U : O_U$, or at attribute value level (or attribute type level), e.g., $x{:}O_i \leq y{:}O_U$ (x is semantically subsumed by y), $x{:}O_i \geq y{:}O_U$ (x semantically subsumes y), $x{:}O_i \equiv y{:}O_U$ (x is semantically equivalent to y), $x{:}O_i$

$\neq y{:}O_U$ (x is semantically incompatible with y), $x{:}O_i \approx y{:}O_U$ (x is semantically compatible with y) [7, 21].

We say that a set of ontologies $O_1, \cdots, O_p$ are integrable according to a user ontology O_U in the presence of the semantic correspondences SC if there exist p partial injective mappings $\psi_1, \cdots, \psi_p$ from $O_1, \cdots, O_p$, respectively, to O_U with the following two properties [9, 6]:

(a) For all $x, y \in O_i$, if $x \preceq y$ in O_i then $\psi_i(x) \preceq \psi_i(y)$ in O_U (order preservation property);
(b) For all $x \in O_i$ and $y \in O_U$, if $(x : O_i \text{ op } y : O_U) \in SC$, then $\psi_i(x)$ op y in the ontology O_U (semantic correspondence preservation property).

In general, the set of mappings can be (semi-automatically) inferred from the set of semantic correspondences specified by the user [9].

3.3 Ontology-Extended Data Sources and Mappings Editor

In many practical data integration scenarios, the ontologies associated with data sources are not explicitly specified in a form that can be manipulated by programs. In such cases, it is necessary to make *explicit*, the implicit ontologies associated with the data sources before data integration can be performed. In addition, users need to be able to specify the user ontology and the semantic correspondences between user ontology and data source ontologies (used later to generate a set of semantics preserving mappings). To address this need, we have developed a user-friendly editor for editing data source descriptions (associated with ontology extended data sources) and for specifying the relevant semantic correspondences (a.k.a., interoperation constraints).

The current implementation of our data source editor provides interfaces for:

(a) Defining attribute types or *isa* hierarchies (attribute value taxonomies) or modifying a predefined set of attribute types.
(b) Defining the schema of a data source by specifying the names of the attributes and their corresponding types.
(c) Defining semantic correspondences between ontologies associated with the data sources and the user ontology.
(d) Querying distributed, semantically heterogeneous data sources and retrieving and manipulating the results according to the user-imposed semantic relationships between different sources of data.

Figure 3 shows the interface that allows specification of semantic correspondences between two data sources. The leftmost panel shows an ontology extended schema associated with a data source, which includes the hierarchical type ontologies associated with attributes. The second panel shows the available semantic correspondences. The third panel shows the ontology extended schema associated with the user data. The user can select a term in the first schema, the desired semantic correspondence, and a term in the second schema. The user-specified semantic correspondences that are used to infer consistent

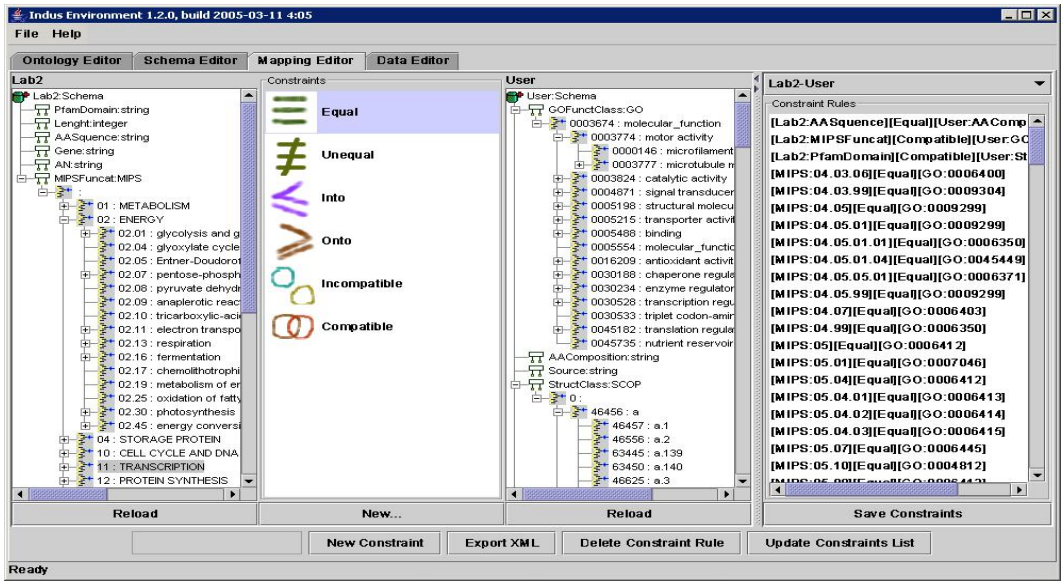

Fig. 3. Editor for defining ontology-extended data sources and semantic correspondences between two ontology-extended data sources.

mappings-specified are shown on the rightmost panel. The ontologies and mappings defined using the user-friendly editor in INDUS are stored in a repository that is available to the query answering engine. INDUS contains a list of predefined mappings (e.g., mappings from *EC Number* to *GO Function* or from *AA Sequence* to *AA Composition*). Some of these functions are procedural (e.g., procedure that maps an *AA Sequence* to *AA Composition*), others represent the enumeration of a list of mappings between values (e.g., *EC Number* to *GO Function*). Furthermore, the user is given the freedom to define new mappings or modify the existing ones according to his or her own needs. For example, if the user wants to map *AA Sequence* to *AA Composition* and this mapping does not exist in the repository, then the user can easily upload the corresponding procedure through the editor interface. Also if a user decides to use a modified version of a pre-defined mapping function, that particular function can be loaded into the editor from the repository and edited according to the user needs.

4 Learning Classifiers for Assigning Protein Sequences to Gene Ontology Functional Families

Caragea et al. [10] have shown that the problem of learning classifiers from distributed data can be reduced to the problem of answering queries from distributed data by decomposing the learning task into an information integration

component in which the information needed for learning (i.e., *sufficient statistics*) is identified and gathered from the distributed data and a hypothesis generation component, which uses this information to generate a model.

Assigning putative functions to novel proteins and the discovery of sequence correlates of protein function are important challenges in bioinformatics. In what follows, we will show how a biologist interested in learning models for predicting the *GO Function* of unlabeled proteins based on data coming from SWISSPROT and MIPS databases, can use the tools provided by INDUS to achieve this task.

4.1 Data and Problem Specification

We consider again the data sources described in our illustrative example. Because the user is interested in learning to predict the *GO Function* of a protein based on the information contained in the amino acid sequence, the data of interest to the user can be seen as coming from two horizontal fragments as in Table 5 (where the data set D_1 is assembled from SWISSPROT and the data set D_2 is assembled from MIPS).

Table 5. Horizontal data fragments that are of interest to a biologist

	Protein ID	*Protein Sequence*	*EC Number*
D_1	P35626	MADLEAVLAD VSYLMAMEKS ···	2.7.1.126 Beta-adrenergic...
	Q12797	MAQRKNAKSS GNSSSSGSGS ···	1.14.11.16 Peptide-aspartate...
	···	···	···
	AC	*AA Sequence*	*MIPS Funcat*
D_2	P32589	STPFGLDLGN NNSVLAVARN ···	16.01 protein binding
	P07278	VSSLPKESQA ELQLFQNEIN ···	16.19.01 cyclic nucleotide bind.
	···	···	···

Typically a user (e.g., a biologist) might want to infer probabilistic models (e.g., Naive Bayes) from the available data. Using INDUS the user defines the semantic correspondences between the data source attributes *Protein ID* in D_1, AC in D_2 and the user attribute ID; *Protein Sequence* in D_1, AA *Sequence* in D_2 and *Sequence* in O_U; and *EC number* in D_1, *MIPS catfun* in D_2 and *GO Function* in the user perspective. Furthermore, the user can use predefined mappings between the values of semantically similar attributes (e.g., mappings from *EC Number* and *MIPS Funcat* to *GO function*) or modify existing mappings according to the user's view of the domain.

We will briefly review the Naive Bayes model, identify sufficient statistics for learning Naive Bayes models from data and show how these sufficient statistics can be computed from distributed, heterogeneous data using INDUS query answering engine.

4.2 Classification Using a Probabilistic Model

Suppose we have a probabilistic model α for sequences defined over some alphabet Σ (which in our case is the 20-letter amino acid alphabet). The model α specifies for any sequence $\overline{S} = s_1, \cdots, s_n$ the probability $P_\alpha(\overline{S} = s_1, \cdots, s_n)$ according to the probabilistic model α. We can construct such a probabilistic model and explore it as a classifier using the following (standard) procedure:

- For each class c_j train a probabilistic model $\alpha(c_j)$ using the sequences belonging to class c_j.
- Predict the classification $c(\overline{S})$ of a novel sequence $\overline{S} = s_1, \cdots, s_n$ as given by: $c(\overline{S}) = \arg\max_{c_j \in C} P_\alpha(\overline{S} = s_1, \cdots, s_n | c_j) P(c_j)$

The Naive Bayes classifier assumes that each element of the sequence is independent of the other elements given the class label. Consequently, $c(\overline{S}) = \arg\max_{c_j \in C} P_\alpha \prod_{i=1}^{n} P_\alpha(s_1|c_j) \cdots P_\alpha(s_n|c_j) P(c_j)$. Note that the Naive Bayes classifier for sequences treats each sequence as though it were simply a bag of letters and it calculates the number of occurences $\sigma(s_i|c_j)$ of each letter in a sequence given the class of the sequence as well as the number of sequences $\sigma(c_j)$ belonging to a particular class c_j. These frequency counts completely summarize the information needed for constructing a Naive Bayes classifier, and thus, they constitute *sufficient statistics* for Naive Bayes classifiers [10]. An algorithm for learning probabilistic models from data can be described as follows:

(1) Compute the frequency counts $\sigma(s_i|c_j)$ and $\sigma(c_j)$.
(2) Generate the probabilistic model α given by these frequency counts.

The query answering engine receives queries such as $q(\sigma(s_i|c_j))$ and $q(\sigma(c_j))$ asking for frequency counts, it decomposes them into subqueries $q_k(\sigma(s_i|c_j))$ and $q_k(\sigma(c_j))$ according to the distributed data sources D_k $(k = 1, p)$ and maps them to the data source ontologies. Once the individual results are received back, the query answering engine composes them into a final result by adding up the counts returned by each data source. Thus, there is no need to bring all the data to a central place. Instead queries are answered from distributed data sources viewed from a user's perspective.

Experimental results on learning probabilistic models for assigning protein sequences to gene ontology functional families are reported by our group in [1]. They show that INDUS can be successfully used for integrative analysis of data from multiple sources needed for collaborative discovery in computational biology.

5 Summary, Discussion and Further Work

5.1 Summary

We have presented INDUS, a federated, query-centric approach to answering queries from distributed, semantically heterogeneous data sources. INDUS assumes a clear separation between data and the semantics of the data (ontologies)

and allows users to specify ontologies and mappings between data source ontologies and user ontology. These mappings are stored in a mappings repository to ensure their re-usability and are made available to a query answering engine. The task of the query answering engine is to decompose a query posed by a user into subqueries according to the distributed data sources and compose the results into a final result to the intial user query.

In previous work [10] we have shown that learning algorithms can be decomposed into an information extraction component and a hypothesis generation component. This decomposition makes it possible to see learning algorithms as pseudo-users that pose queries to the query answering engine in order to gather the information that they need for generating the models that they output. Modular implementations of several learning algorithms have been linked to INDUS, thus obtaining algorithms for learning classifiers from distributed, semantically heterogeneous data sources. We have demonstrated how we can use INDUS to obtain algorithms for learning Naive Bayes models for predicting the functional classification of a protein based on training sequences that are distributed among several distributed, semantically heterogeneous data sources.

An initial version of INDUS software and documentation are available at: `http://www.cild.iastate.edu/software/indus.html`.

5.2 Discussion

There is a large body of literature on information integration and systems for information integration. Davidson et al. [12] and Eckman [13] survey alternative approaches to data integration. Hull [19] summarizes theoretical work on data integration. Several systems have been designed specifically for the integration of biological data sources. It is worth mentioning SRS [15], K2 [29], Kleisli [11], IBM's DiscoveryLink [18], TAMBIS [28], OPM [22], BioMediator [25], among others.

Systems such as SRS and Kleisli do not assume any data model (or schema). It is the user's responsability to specify the integration details and the data source locations, when posing queries. Discovery Link and OMP rely on schema mappings and the definition of views to perform the integration task. TAMBIS and BioMediator make a clear distinction between data and the semantics of the data (i.e., ontologies) and take into account semantic correspondences between ontologies (both at schema level and attribute level) in the process of data integration.

Most of the above mentioned systems assume a predefined global schema (e.g., Discovery Link, OMP) or ontology (e.g., TAMBIS), with the notable exception of BioMediator, where users can easily tailor the integrating ontology to their own needs. This is highly desirable in a scientific discovery setting where users need the flexibility to specify their own ontologies.

While some of these systems can answer very complex queries (e.g., BioMediator), others have limited query capabilities (e.g, SRS which is mainly an information retrieval system). Furthermore, for some systems it is very easy to add new data sources to the system (e.g., SRS or Kleisli, where new data source

wrappers can be easily developed), while this is not easy for other biological integration systems (e.g., Discovery Link or OMP, where the global schema needs to be reconstructed).

Finally, while some systems (e.g., SRS, BioMediator) provide support for biological information retrieval tools (such as BLAST or FASTA), to the best of our knowledge none of them are linked to machine learning algorithms that can be used for data analysis, classification or prediction.

On a different note, there has been a great deal of work on ontology development environments. Before developing INDUS editor, off-the-shelf alternatives such as IBM's Clio [14] or Protege [24] were considered, but they proved insufficient for our needs. Clio provides support only for schema mapping, but not for hierarchical ontology mapping. Protege is a purely knowledge base constructing tool (including ontology mappings). It does not provide support for the association of ontologies with data, data management or queries over the data. Furthermore, neither of these systems allow procedural mappings (a.k.a., conversion functions), which are essential for data integration.

Of particular interest to ontology-based information integration is work on modular ontolgies. Ontolingua [17, 16] and ONION [23] support manipulation of modular ontologies. Calvanese et al. [8] proposed a view-based mechanism for ontology integration. However, a global ontology is typically unavailable in information integration from loosely linked, distributed, semantically heterogeneous data. We have explored a description logic based approach to modular design and reuse of ontologies, specification of inter-ontology semantic correspondences, and mappings [4]. However, support for asserting and reasoning with partially specified semantic correspondences between local ontologies and localized reasoning in distributed description logic is lacking.

In terms of learning from distributed, semantically heterogeneous data, while there is a lot of work on distributed learning (see [20] for a survey), there has been little work on learning classifiers from semantically heterogeneous, distributed data. Ontology extended relational algebra [6] provides a framework within which users can specify semantic correspondences between names and values of attributes and obtain answers to relational queries. This approach has been extended in our work on INDUS to handle more general statistical queries across semantically heterogeneous data sources [9].

5.3 Further Work

Our approach has been applied successfully to scenarios where the ontologies associated with some attributes are given by tree structured *isa* hierarchies. It is desirable to extend our work to the more general case where the hierarchies are directed acyclic graphs, as this case is more often encountered in practice.

As Protege [24] is the most popular tool for creating knowoledge bases, in the future INDUS will allow users to import ontologies that are edited using Protege.

In our current framework, we assume that each data source can be seen as a single table. It is of interest to extend INDUS to scenarios where each data

sources can be conceptually viewed as a set of inter-related (possibly hierarchical) tables. This requires a framework for asserting semantic correspondences between tables and relations across multiple ontologies (see [14]). In this context, recent work on description logics for representing and reasoning with ontologies [3, 27], distributed description logics [7] as well as ontology languages, e.g., web ontology language (OWL) [26] are of interest. These developments, together with our work on INDUS, set the stage for making progress on the problem of integration of a collection of semantically heterogeneous data sources where each data source can be conceptually viewed as a set of inter-related tables in its full generality.

Acknowledgements: This work was funded in part by grants from the National Science Foundation (IIS 0219699) and the National Institutes of Health (GM 066387).

References

1. C. Andorf, A. Silvescu, D. Dobbs, and V. Honavar. Learning classifiers for assigning protein sequences to gene ontology functional families. In *Fifth International Conference on Knowledge Based Computer Systems (KBCS 2004)*, India, 2004.
2. M. Ashburner, C. Ball, J. Blake, D. Botstein, H. Butler, J. Cherry, A. Davis, K. Dolinski, S. Dwight, J. Eppig, M. Harris, D. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J. Matese, J. Richardson, M. Ringwald, G. Rubin, and G. Sherlock. Gene ontology: tool for unification of biology. *Nature Genetics*, 25(1):25–29, 2000.
3. F. Baader and W. Nutt. Basic description logics. In F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors, *The Description Logic Handbook: Theory, Implementation, and Applications*, pages 43–95. Cambridge University Press, 2003.
4. J. Bao and V. Honavar. Collaborative ontology building with wiki@nt - a multi-agent based ontology building environment. In *Proceedings of the Third International Workshop on Evaluation of Ontology based Tools, at the Third International Semantic Web Conference ISWC*, Hiroshima, Japan, 2004.
5. T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, May 2001.
6. P. Bonatti, Y. Deng, and V. Subrahmanian. An ontology-extended relational algebra. In *Proceedings of the IEEE Conference on Information Integration and Reuse*, pages 192–199. IEEE Press, 2003.
7. A. Borgida and L. Serafini. Distributed description logics: Directed domain correspondences in federated information sources. In *Proceedings of the Intenational Conference on Cooperative Information Systems*, 2002.
8. D. Calvanese, G. D. Giacomo, and M. Lenzerini. A framework for ontology integration. In *Proceedings of the international semantic web working symposium*, pages 303–316, Stanford, USA, 2001.
9. D. Caragea, J. Pathak, and V. Honavar. Learning classifiers from semantically heterogeneous data. In *Proceedings of the International Conference on Ontologies, Databases, and Applications of Semantics for Large Scale Information Systems*, 2004.
10. D. Caragea, A. Silvescu, and V. Honavar. A framework for learning from distributed data using sufficient statistics and its application to learning decision trees. *International Journal of Hybrid Intelligent Systems*, 1(2), 2004.

11. J. Chen, S. Chung, and L. Wong. The Kleisli query system as a backbone for bioinformatics data integration and analisis. *Bioinformatics*, pages 147–188, 2003.
12. S. Davidson, J. Crabtree, B. Brunk, J. Schug, V. Tannen, G. Overton, and C. Stoeckert. K2/Kleisli and GUS: experiments in integrated access to genomic data sources. *IBM Journal*, 40(2), 2001.
13. B. Eckman. A practitioner's guide to data management and data integration in bioinformatics. *Bioinformatics*, pages 3–74, 2003.
14. B. Eckman, M. Hernndez, H. Ho, F. Naumann, and L. Popa. Schema mapping and data integration with clio (demo and poster). In *Proceedings of the International Conference on Intelligent Systems for Molecular Biology (ISMB 2002)*, Edmonton, Canada, 2002.
15. T. Etzold, H. Harris, and S. Beulah. SRS: An integration platform for databanks and analysis tools in bioinformatics. *Bioinformatics Managing Scientific Data*, pages 35–74, 2003.
16. R. Fikes, A. Farquhar, and J. Rice. Tools for assembling modular ontologies. In *The Fourteenth National Conference on Artificial Intelligence*, 1997.
17. T. Gruber. Ontolingua: A mechanism to support portable ontologies.
18. L. Haas, P. Schwarz, P. Kodali, E. Kotlar, J. Rice, and W. Swope. DiscoveryLink: a system for integrated access to life sciences data sources. *IBM System Journal*, 40(2), 2001.
19. R. Hull. Managing semantic heterogeneity in databases: A theoretical perspective. In *PODS*, pages 51–61, Tucson, Arizona, 1997.
20. H. Kargupta and P. Chan. *Advances in Distributed and Parallel Knowledge Discovery*. AAAI/MIT, 2000.
21. A. Kementsietsidis, M. Arenas, and R. J. Miller. Mapping data in peer-to-peer systems: Semantics and algorithmic issues. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 325–336, 2003.
22. A. Kosky, I. Chen, V. Markowitz, , and E. Szeto. Exploring heterogeneous biological databases: Tools and applications. In *Proceedings of the 6th International Conference on Extending Database Technology (EDBT98), Lecture Notes in Computer Science Vol. 1377, Springer-Verlag*, 1998.
23. P. Mitra, G. Wiederhold, and M. Kersten. A graph-oriented model for articulation of ontology interdependencies. In *Conference on Extending Database Technology*, Konstanz, Germany, 2000.
24. N. F. Noy, R. W. Fergerson, and M. A. Musen. The knowledge model of protege-2000: Combining interoperability and flexibility. In *Second International Conference on Knowledge Engineering and Knowledge Management (EKAW'2000)*, Juan-les-Pins, France, 2000.
25. R. Shaker, P. Mork, J. S. Brockenbrough, L. Donelson, and P. Tarczy-Hornoch. The biomediator system as a tool for integrating biologic databases on the web. In *Proceedings of the Workshop on Information Integration on the Web (held in conjunction with VLDB 2004)*, Toronto, ON, 2004.
26. M. Smith, C. Welty, and D. McGuinness. *OWL Web Ontology Language Guide*. W3C Recommendation, 2004.
27. S. Staab and R. Studer. *Handbook on Ontologies*. International Handbooks on Information Systems Springer, 2004.
28. R. Stevens, C. Goble, N. Paton, S. Becchofer, G. Ng, P. Baker, and A. Bass. Complex query formulation over diverse sources in tambis. *Bioinformatics*, pages 189–220, 2003.
29. V. Tannen, S. Davidson, and S. Harker. The information integration in K2. *Bioinformatics*, pages 225–248, 2003.