

Knowledge and Data Integration for Discovery

John Aronis¹ Carolina Ruiz^{2,1} Bruce Buchanan¹
aronis@cs.pitt.edu ruiz@cs.wpi.edu buchanan@cs.pitt.edu

¹ Department of Computer Science
University of Pittsburgh
Pittsburgh, PA 15260

² Department of Computer Science
Worcester Polytechnic Institute
Worcester, MA 01609

1 Introduction

As organizations collect ever larger databases, they are beginning to realize the importance of *mining* the accumulated wisdom inherent in their data archives. Data mining is a new field with exciting early successes. These successes, however, are tempered by the difficulties involved in mining large, complex data sources across an organization. Furthermore, although the importance of background knowledge has been amply demonstrated, data mining methods are generally unable to make use of it on a large scale. We describe here a prototype system for preprocessing data in the context of background knowledge.

Data mining is an iterative process: first, the user selects and prepares parts of the available data, then applies various machine learning and discovery tools to it, analyzes the results, then iterates the process over. Although running the mining tools and analyzing the results might be the most interesting parts of this process, selecting and preparing the data are often the most difficult and time-consuming. This *preprocessing* phase can be complex—in addition to mechanical operations to reformat and standardize databases, background knowledge is often incorporated into the data during this stage, and the process is often complicated by the need to work with heterogeneous databases. Furthermore, because of the iterative nature of data mining, the preprocessing phase may be repeated many times, making changes and incorporating the results of each mining operation into the next cycle.

This paper describes a prototype system that semi-automatizes the preprocessing phase. This system allows miners to work within a logically defined abstraction space of concepts, definitions, and previously learned rules to select and prepare data for each cycle of the mining process. In particular, it enables them to define *background knowledge* about databases and domains in a simple, declarative language, and to incorporate it into the discovery process. It also allows users to incrementally refine this background knowledge with new terminology and definitions on each cycle, and to reuse it in future mining tasks.

2 A Deductive Approach to Data Mining

We have identified three major categories of data manipulation operations. Although this classification is somewhat arbitrary, it accounts for much of the work spent preparing data for mining, and provides a convenient organization:

Integrating Databases. This includes normalizing databases, integrating heterogeneous data sources, fixing missing and corrupted values, enforcing integrity constraints, and standardizing usage of terms and values.

Incorporating Domain Knowledge. Domain knowledge may include general categories and relationships, background databases, and “textbook” definitions of concepts and quantities.

Constructing new features. This consists of generating new terminology and definitions from existing features and domain knowledge.

These operations are generally implemented in a procedural fashion, with ad hoc programs to transform the original data into a form suitable for mining. As the mining operation progresses, parts of the code must be modified and rerun to produce new datasets.

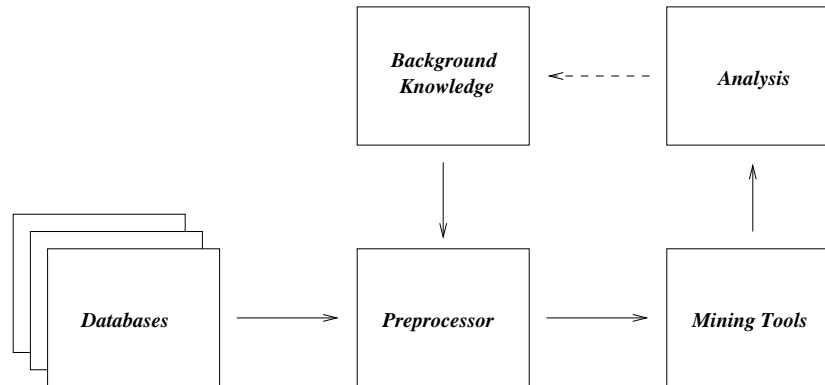


Figure 1: System Architecture.

We have designed and implemented a system that allows background knowledge and transformations to be stated declaratively in a unified format. The *background knowledge* consists of three main components corresponding to the major classes of preprocessing operations mentioned above. First, a collection of statements about database formats and fields, and definitions of codes and their usage. Second, a body of domain knowledge, including taxonomical hierarchies and constraints, represented in a declarative language. Third, a group of definitions of new quantities and concepts to be used as the mining operation progresses.

One of the advantages of this approach is that a declarative description of background information is easy to understand, simple to update and change, and suitable to be incrementally enriched with information discovered as the mining operation progresses. An evolving set of “domain knowledge files” can be shared by several people working on a set of related problems; more general aspects of background knowledge, such as unit conversions and basic definitions, can be stored in a library for general use.

3 An Illustration

The system supports the iterative process of data mining. The miner generally starts the process by specifying file formats and field conversions, and defining the basic terminology used in the data. Then, on each iteration of the process, the miner uses concepts defined in the background knowledge to select and prepare data for mining. After analyzing the results of each mining operation, the user can update the background knowledge, and repeat the cycle. We illustrate the use of the system in the domain of learning characteristics of protein binding sites. One of the first objectives of this project is to learn to recognize calcium-binding (Ca^{2+}) sites in protein structures.

A *site* is a 3-dimensional region around a particular atom. The site can be characterized by the atoms and structures present in various concentric *shells* around the central atom. Information about sites is given in two files which are loaded with the commands:

```
load sites.db delimiters: cr/tab fields: name center binding-site
load atoms.db delimiters: cr/tab fields: in-site type x y z
```

The `sites.db` file contains for each site its identifier, the location of its center, and whether or not it is a calcium-binding site (i.e. whether the site is a positive or a negative example). The `atoms.db` file contains a list of the atoms (carbons, nitrogens, oxygens, or others) belonging to each site, together with the 3-dimensional location (x, y, z coordinates) of each atom.

After the basic data points have been loaded into the system, the miner begins to define the concepts and terminology of the domain. The `properties.db` file contains a classification of the atoms according to a collection of user-defined properties and background information (see) including: chemical groups (such as hydroxyl, amide, amine, carbonyl, ring-system, peptide), residue-based properties (such as residue types and hydrophobicity), secondary structure, and biophysical characteristics (such as charge, polarity, mobility, and solvent accessibility).

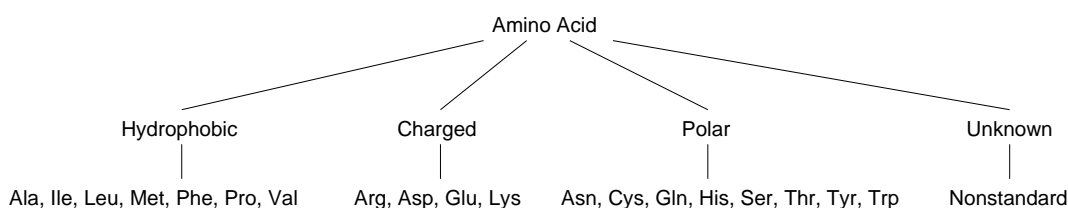


Figure 2: Hydrophobicity Classification I.

For brevity, in the sequel we consider only two taxonomies. of atomic properties. One classification of the 20 common amino acids (and correspondingly of the atoms occurring in them) according to their hydrophobicity (hydrophobic/charged/polar/unknown), and an alternate classification of the hydrophobicity values (polar/non polar and acidic/basic).

Our preprocessor brings together all information about each site (and about each atom) which is originally distributed across different files into a set of predicates that work together. As mentioned before, this information is stored in a declarative manner. To illustrate this, assume the file `sites.db` contains a positive example of a calcium-binding site `cb1`. Assume

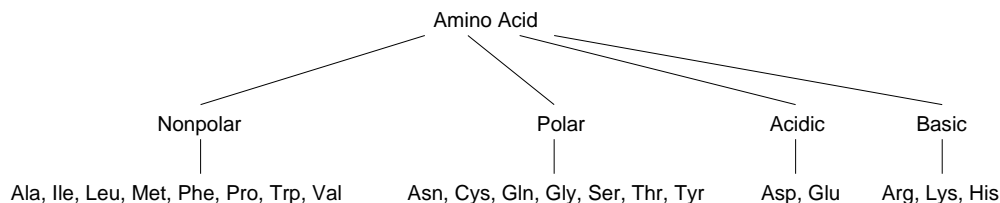


Figure 3: Hydrophobicity Classification II.

also that it consists of three atoms (which is greatly simplified): an oxygen atom `oxy1`, which is part of a Glutamine (Gln) amino acid, a nitrogen atom `nit1`, which is part of a Histidine (His) amino acid, and a carbon `carb1`, which is part of a Phenylalanine (Phe) amino acid. The load command stores this information about `cb1` using the following predicates:

```

site(cb1)           binding-site(cb1,yes)
has-atom(cb1,oxy1)  isa(oxy1,oxygen)       in-amino-acid(oxy1,gln)
has-atom(cb1,nit1)  isa(nit1,nitrogen)    in-amino-acid(nit1,his)
has-atom(cb1,carb1) isa(carb1,carbon)     in-amino-acid(carb1,phe)
  
```

(The miner does not name each atom, rather unique names are assigned by the load procedure.) Even though the properties of each individual atom are not explicitly specified in any of the original three databases, our system deduces them from the background knowledge.

The main procedure that drives the rest of the mining process is MINE. The miner uses it to select a subset of the data for mining, select a target concept, and create the necessary files for mining. A simple, but typical invocation of MINE looks like:

```

MINE Item SATISTYING site(Item)
  CONCEPT          VALUE-OF X IN binding-site(Item,X)
  NUMBER-IN-SHELL COUNT-OF X IN [in-site(X,Item) AND in-sphere(X,2,0,0,0)]
  
```

Data are selected on the first line as the variable `Item` is declared to range over the extension of the predicate `site`. The procedure creates a table with two fields:

- **CONCEPT** specifies the target concept for the mining tool to learn. The **VALUE-OF** keyword says to use the value of the `binding-site` predicate for each value of `Item`.
- **NUMBER-IN-SHELL** This field is defined by the second-order operator `COUNT` which counts how many objects satisfy the predicate.

In practice, the mining operation must be repeated many times while adjusting various parameters.

4 Proposed Work

We propose to extend this work to the development of tools for the advanced analysis of large datasets, including heterogeneous and distributed databases. In particular, we propose: described here by:

- To develop a library of feature extraction routines to be used with the preprocessor for molecular biology problems.
- To develop inter-databases mediators that can link heterogeneous databases for use in joint learning problems.
- To build a robust set of tools for theory formation and model building, particularly in the molecular biology and clinical medicine domains.

The preprocessor described here facilitates much of the scientific discovery process; we propose to automate larger parts of it. We envision a system for collaborative man/machine model building and theory formation. The system will perform many of the tasks now performed by the scientist using its own control structure. The scientist is still free to participate, to modify the behavior of the system, or simply use the results. We expect the full development of such a system will take three to five years.

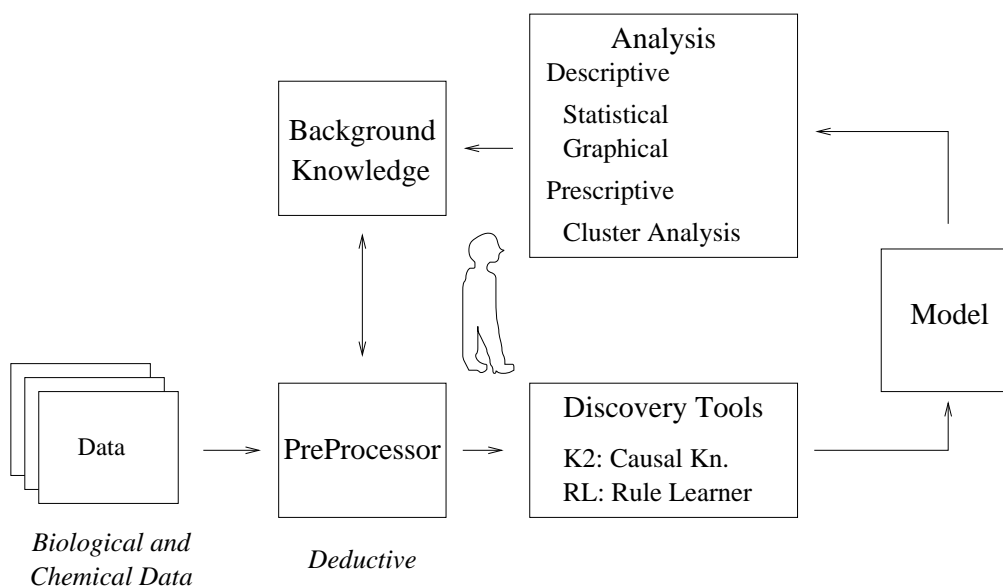


Figure 4: Model Building.

5 Personnel

Bruce G. Buchanan Dr. Buchanan received a BA (Mathematics, 1961) from Ohio Wesleyan University and PhD degree (philosophy, 1966) from Michigan State University. Since joining the University of Pittsburgh faculty in 1988 he has been co-director of the Center for Parallel, Distributed, and Intelligent Systems, a fellow in the Center for Philosophy of Science, and co-director of the Keck Center for Computational Biology. He was named a University Professor in 1993. He has joint appointments with the Departments of Philosophy and Medicine, and with the Intelligent Systems Program. Dr. Buchanan's main research interest is artificial intelligence, in particular, intelligent computer methods for knowledge

acquisition and machine learning, scientific hypothesis formation, and construction of expert systems for scientific problems. He was one of the principals in the design and development of DENDRAL, Meta-DENDRAL (RL), MYCIN, E-Mycin, and PROTEAN systems. He is a fellow of the American Association for Artificial Intelligence, a fellow of the American College of Medical Informatics, and is on the editorial boards of the journals *Artificial Intelligence*, *Machine Learning*, *Expert Systems*, *Knowledge Acquisition*, and *Computational Molecular Cell Biology*.

John M. Aronis Dr. Aronis has been working with Dr. Buchanan on inductive learning since he received his Ph.D. under Dr. Richmond Thomason at the University of Pittsburgh. He is involved in all aspects of inductive learning from theory to applications, and has been a positive force in moving the RL program toward firmer grounding in logical theory.

Carolina Ruiz Dr. Ruiz is a recent graduate in Computer Science under Dr. Jack Minker at the University of Maryland. Her dissertation involves formalizing negations of queries in databases. She has spent the 1996-97 academic year in Dr. Buchanan's laboratory working on formalizing the links between databases and automated discovery systems. She will continue this work when she assumes her faculty position at Worcester Polytechnic Institute in the fall of 1997.