Biological data cleaning: a case study

Katherine G. Herbert*

Department of Computer Science,
Montclair State University,
Montclair, New Jersey, 07043, USA
E-mail: herbertk@mail.montclair.edu
*Corresponding author

Jason T.L. Wang

Bioinformatics Center and Department of Computer Science,
New Jersey Institute of Technology,
University Heights, Newark, NJ 07102, USA
E-mail: wangj@njit.edu

Abstract: As databases become more pervasive through the biological sciences, various data quality concerns are emerging. Biological databases tend to develop data quality issues regarding data legacy, data uniformity and data duplication. Due to the nature of this data, each of these problems is non-trivial and can cause many problems for the database. For biological data to be corrected and standardised, methods and frameworks must be developed to handle both structural and traditional data. This paper discusses issues concerning biological data quality with respect to data cleaning. It presents BIO-AJAX, a framework developed to address these issues. It finally describes BIO-JAX for TreeBASE and BIO-AJAX for Lineage Path, two implementations of BIO-AJAX on phylogenetic data sets.

Keywords: bioinformatics; biological data quality; data quality; data cleaning; information quality; phylogenetic data.

Reference to this paper should be made as follows: Herbert, K.G. and Wang, J.T.L. (2007) 'Biological data cleaning: a case study', *Int. J. Information Quality*, Vol. 1, No. 1, pp.60–82.

Biographical notes: Katherine G. Herbert is an Assistant Professor of Computer Science at Montclair State University in New Jersey. She completed her BS Degrees in Mathematics and Computer Science at Saint Peter's College and her MS and PhD Degrees at the New Jersey Institute of Technology under the supervision of Jason T.L. Wang. At Montclair State University, she has created and is Director of the Biological Data Quality Lab, supporting research for Computer Science, Information Technology and Science Informatics undergraduate and graduate students.

Jason T.L. Wang received the BS Degree in Mathematics from National Taiwan University, Taipei, Taiwan, and the PhD Degree in Computer Science from the Courant Institute of Mathematical Sciences, New York University, in 1991. He is a Full Professor of Computer Science in the College of Computing Sciences at New Jersey Institute of Technology and Director of the university's Data and Knowledge Engineering Laboratory. His research interests include data mining and databases, bioinformatics, and cyberinfrastructure. He has

published over 120 refereed journal and conference papers in these areas. He is co-author of the book *Mining the World Wide Web: An Information Search Approach* (2001, Kluwer/Springer), and an Editor and author of three books *Pattern Discovery in Biomolecular Data: Tools, Techniques and Applications* (1999, Oxford University Press), *Computational Biology and Genome Informatics* (2003, World Scientific), and *Data Mining in Bioinformatics* (2005, Springer). He is the Executive Editor of World Scientific Book Series on Science, Engineering, and Biology Informatics (SEBI), and serves on the editorial boards of five journals including *Information Systems*, *Knowledge and Information Systems*, *Intelligent Data Analysis*, *International Journal of Data Mining and Bioinformatics*, and *International Journal of Information Quality*.

1 Introduction

When most databases are designed, the schema is perfected and issues concerning the data are taken into account. However, once a database becomes active, a number of situations can occur that may necessitate additional tools to either improve or maintain the integrity of the data. If knowledge about the data changes quickly, this can result in legacy data not conforming to the knowledge contained in more recent data. Moreover, if databases are transferred into data warehouses or if it becomes a part of a federated database system, the data within the system must be integrated to conform to one schema.

The areas of data quality, cleaning and integration discuss the aforementioned issues concerning databases. Data quality primarily concerns itself with issues concerning the characteristics of a database's data and schema. It then analyses whether the actual database conform to the expected view of the data from the conceptual model. Data quality usually addresses where or not the records within the database are accurate, timely, complete and consistent (Wang et al., 1995). Three methods for managing data quality are data cleaning (Rahm and Do, 2000), data quality monitoring and data integration (Lenzerini, 2002). Primarily, data cleaning addresses the issues of "detecting and removing errors and inconsistencies from data in order to improve the quality of the data" (Lenzerini, 2002).

This paper will discuss the need for biological data cleaning and present framework for performing data cleaning on biological databases. It will demonstrate how this framework, BIO-AJAX (Herbert et al., 2004), has been applied effectively to phylogenetic data, a heterogeneous and complex data set containing both structural and traditional text data through BIO-AJAX for TreeBASE and BIO-AJAX for Lineage Paths. While the process has been addressed in the general case theoretically and analysed for specific databases, there has been very little work regarding data cleaning when referring to biological and evolutionary databases. Finally, it will address some open research questions and conclude with describing future work.

1.1 Motivations for biological data cleaning

Biological data is rich with issues that can be addressed with data cleaning and integration methodologies. Data cleaning in biological data is an important function necessary for the analysis of biological data. It can standardise the data for further

computation and improve the quality of the data for searching. The very core purpose for most biological databases is to create repository, integrating work from numerous scientists. Phylogenetic data encapsulates these problems. Using phylogenetic data to demonstrate these problems, data cleaning and integration can be used to solve the following problems:

- standardising the format of phylogenetic data when performing operations upon them
- cleaning and modifying legacy data
- standardising nomenclature
- finding duplicate structural data (trees) and records within the dataset
- removing duplicates, if necessary
- clustering similar structural data (trees) and records
- merging similar records
- finding anomalous structural data (trees) and data.

Each of these problems can pose serious problems for phylogenetic databases. First, many databases have standardisation issues. If the data is manually entered into the database, variations in how the user inputs the data can cause problems. If the database contains legacy data, there can also be standardisation problems. With phylogenetic and biological data, innovations and discoveries within the field occur rapidly. Therefore the knowledge about the data, as well as how different data affects other data can change just as quickly. To reflect these changes, data may be stored differently, with the data that did not benefit from these new discoveries, kept in its same format. With data now in at least two, possibly multiple formats, it becomes harder to apply any computational tools to this data or even perform effective retrieval. Therefore, it becomes imperative to integrate the data in these various formats.

Another large problem facing phylogenetic databases is the nomenclature issue. There are various types of nomenclature systems, but with phylogenetic data, the primary nomenclature issues concern the naming schemes related to how the species are classified. Currently, the pervasive nomenclature methodology within biology for species is the Linnaean Hierarchy. Many phylogenetic researchers find this system of classification inadequate and have proposed other nomenclature methods (Soares, 2004). The Linnaean Hierarchy primarily depends upon naming species based on the observations. With the advent of computation biology and more quantitative methods for determining the relationships between species, differences in how to classify species have arisen. Since the Linnaean Hierarchy is dependent upon the hierarchical classification for naming, and since there are now multiple methods for creating the hierarchy, many phylogenetic researchers feel the hierarchy is inadequate. This has resulted in the creation and debate over a number of mechanisms for naming species. Many of these nomenclature systems are in development. Moreover, concerning legacy data, nomenclature rules were not followed perfectly while naming species. While better-known species such as Homo sapiens (human) have standard nomenclature, many obscure species, such as species within the botanical fields, can have nomenclature issues. Synonymy occurs since many of the botanical species were named through visual

inspection rather than using modern computational methods. Therefore, a species can have two distinct Linnaean names. By using Linnaean names exclusively, this can limit the users of a phylogenetic database tool. Many potential users, possibly students or recreational users, may be unfamiliar with the various scientific names associated with an species. This creates the problem of standardising nomenclature for performing database operations such as searching while also needing to maintain flexibility with the database by allowing the user to specify the nomenclature of choice. It also creates interesting cleaning problems for matching (Harlin, 2003).

Issues can also exist in phylogenetic databases regarding duplicate structure or text-based data. In this type of database, duplicates need to be handled very carefully. A duplicate tree may not indicate a duplicate record. It can indicate duplicate findings by two different studies or a continuation of a previous study. This type of information can be of interest to both the database curator and the user since it does give information about related trees. By detecting duplicates or trees that are extremely similar, the database can then give this information to the user. The user can then possibly learn about the differences in the trees or the records containing the trees. These differences can be of importance. For example, if two trees are similar but were developed by two different research teams using two different methods for constructing the trees, this could be of importance since it shows similar results by two separate methods. If two records are deemed duplication errors, then duplicate detection can also eliminate the duplicate record.

Clustering phylogenetic data, like any other data, can be very useful. Generally, clustering data allows for patterns within the data to be found. Based on the measure used for clustering, similar records (or in this case phylogenetic trees) can be grouped together. This leads to interesting possibilities concerning the analysis of the clusters. First, the clusters can give information on what trees are similar. Clustering can be based on structure, method of creation of the tree, similar species within the tree as well as other possible parameters. This information can be of use to both the curators and the users. The curators can use this information to study the behaviour of the database. It can also be used as a method of detecting errors, since if a tree is an outlier within the clustering, and after analysis it is found the tree should be within a specific cluster, the curator can then examine the specific tree for possible errors or other automated tools can be applied to the outliers to confirm it should be an outlier. Also, clusters can also create preliminary groupings of trees so that data can be merged.

Each of these cleaning and integration issues has long reaching effects in phylogenetic research. Since the quality of the data is improved through the data cleaning, recall and precision can be improved upon the database. Also, any other computational tool applied to the data within the database performs better. Since the data is clean, the tool's performance is based more on the content of the data. Without the cleaned data, the tool's results can be skewed and reflect the errors and abnormalities within the database.

All of these problems, while specifically belonging to phylogenetic data, particularly data housed in TreeBASE, http://www.treebase.org (Piel et al., 2000), a manually curated phylogenetic tree repository, can be generalised to biological databases. The BIO-AJAX framework was developed and applied to many of the issues observed within TreeBASE. While this framework is primarily applied to phylogenetic data, it can also be applied to other biological data sets. For example, while TreeBASE does not automatically exchange data with other databases, a biological data cleaning scheme can help clean

problems associated with that type of database operation through mapping schema properly or standardising the nomenclature between the two databases. Moreover, mapping can be interpreted in terms of data integration. Also, it can help a stand alone database address its nomenclature problems. Nomenclature problems are pervasive among biological data sets. Some other nomenclature problems are a result of nucleotide research while others discuss protein data (Greer et al., 2002). A data cleaning framework can also address specific error detection issues inherent to a particular biological data set. It can also cluster the data on metrics specifically associated with the data to perform error detection and preprocessing for merging.

This article discusses the aforementioned problems and introduces new methods to help preserve biological data quality. Section 3 reviews the state of the art in data cleaning, a key component of data quality. Section 4 introduces BIO-AJAX and formalises the conceptual framework. Section 5 demonstrate practical implementations of BIO-AJAX. It exhibits BIO-AJAX for TreeBASE, a phylogenetic nomenclature cleaning tool that employs mediator integration. It also presents BIO-AJAX for Lineage Paths, a data quality tool that uses data warehousing techniques to allow users to manipulate taxonomic lineage paths from the NCBI Taxonomy Database and the Integrated Taxonomic Information Server. Finally, Section 6 discusses the implications these biological data quality methods will have for data repositories and their information retrieval and knowledge discovery tools. It will also discuss possible future projects concerning data quality especially concerning BIO-AJAX.

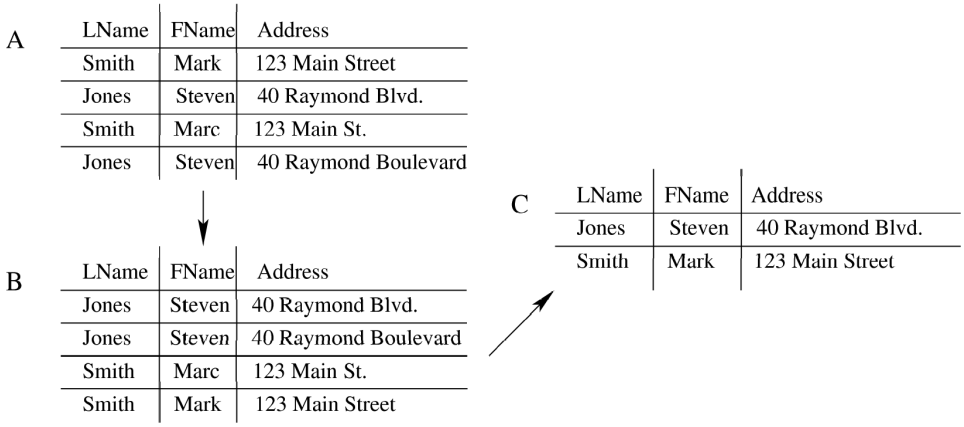
2 General data cleaning

As mentioned previously, data cleaning methods address the issues concerning the quality of data (Dasu and Johnson, 2002, 2003). It is also sometimes referred to as the process of taking data that is 'noisy', sometimes also referred to as 'dirty', or otherwise in a format where computational tools have difficulty processing it and transforming it into a more accessible format. This process maintains the integrity of the data while removing aspects of this data that can be considered 'dirty' (Dasu and Johnson, 2002, 2003).

Data cleaning, which is a part of a larger research area called 'data quality', is a very large field that encompasses a number of research areas within database. In data cleaning, there are three categories of problems that need to be dealt with. First, there is erroneous data detection. Problems from erroneous data usually stem from, but are not limited to, user input errors, inconsistency in input, missing values, misspellings, improper generation of data, and legacy data difference (Dasu and Johnson, 2002, 2003).

Next, there is duplicate detection. Figure 1 demonstrates a simplified version of this method. At first, duplicate detection was applied to very large databases where duplication control was not very strong. The first algorithms were essentially string detection and record detection algorithms for relational databases. However, as data and schemas grew more complex, duplication detection became harder. Also the questions concerning whether two similar documents were actually duplicated became a pressing question. With simple similarity detection through sorting and joining records within a database, more complex duplication errors became evident. Duplication can occur, but not be detected, through data errors. Also, as data becomes more complex, synonymy can occur. Synonymy occurs when two records are not identical syntactically, but are identical semantically (Rahm and Do, 2000).

Figure 1 Data cleaning through duplicate detection. 'A' represents the unclean database. 'B' shows the database after sorting and detecting duplicates. 'C' presents the final database



Finally, there is schema errors and schema and data integration. Schema errors result from improperly formed schema or schema that need to change frequently. Schema and data integration occurs when the database is part of a federated system or needs to be synchronised with other databases. With interoperability being facilitated through the World Wide Web, many databases tend to need to synchronise with other databases. This process of synchronisation includes communicating both data and schema between databases and updating the database accordingly. Therefore, proper schema becomes essential for communicating with other database. Moreover, for rapidly changing data, schemas sometimes need to be modified to reflect these changes in data. Without properly mapping previous data, the database can have trouble with legacy data interacting with new data (Rahm and Do, 2000).

Data cleaning problems have been observed throughout the history of databases, with the first documented cases stemming from the US Census information during the 1950s. Since then, there have been many developments within the data-cleaning field. Commonly, a data cleaning system should fulfil the following conditions. First, it should remove errors within the data. This includes errors within the data as well as schema integration issues for federated databases. Second, the cleaning should be done through automated tools rather than manual inspection. Third, data cleaning should not be done in isolation but rather with respect to the entire database and its schema (Rahm and Do, 2000). Table 1 summarises data cleaning methods and gives an example for each method.

2.1 Duplicate elimination

One of the earliest methods for cleaning data was duplicate detection and elimination. One of the most popular method for duplicate detection was introduced by Bitton and DeWitt (1983). At the time, for most database management systems, trivial duplicates could be found through sorting and then performing a join procedure. The join would then find the duplicates and they could be eliminated. However, this was a time-intensive procedure. Many database management systems chose not to sort in favour of speed. The method developed by Bitton and DeWitt (1983) modifies the sort procedure so that it

ran somewhat faster than the sort procedure provided by the relational databases of the time. Also, this sort procedure, upon detecting a duplicate, immediately removed the duplicate, thereby eliminating the need for the join procedure. The sorting algorithm can be any efficient sorting algorithm that allows for external sorting. Some of these algorithms include two-way merge sort and the use of hash tables (Bitton and DeWitt, 1983).

Since the Bitton method has been introduced, extensive work has been done to detect duplicates and approximate duplicates within databases. One of the major advancements in this type of approach to duplicate detection is the sorted neighbourhood approach to detecting duplicates. In the sorted neighbourhood approach, a key is extracted from the data. This key, while being a unique identifier for the data, also represents the minimum set of elements from which similarity can be determined. This key is usually a combination of the elements within the given set of attributes rather than all of the data within the attributes. This increases the speed of the sorting as well and gives a smaller string for comparison. Records are then sorted based on this key. From this sorted list of records, similarity can be measured through the use of a 'window'. This window W represents W records from the sorted set. Any record R within W is then compared with the $W-1$ record before R and $W-1$ records after R in sorted order. If the records then are judged to be similar, the duplicates are eliminated (Hernandez and Stolfo, 1995, 1998; Low et al., 2001).

While the sorted neighbourhood method detects many duplicates, it does have its limitations. For example, it only detects duplicates within the given window size. Moreover, duplicate detection is highly depended upon the key selected for sorting the records. Based on these observations, Hernandez and Stolfo (1995, 1998) developed a modified sorted neighbourhood method that tries to improve performance based on the above problem. In their method, they perform the sorting through multiple iterations. During each sort, the key is modified so that the sort results will be different. After each sort, the records are compared, with duplicates being deleted and similar records being merged. However, since there is multiple passes through this sort and merge algorithm, more record that are similar can be detected. Also, relating records that are not obviously similar becomes easier by taking the union of the set of results obtained by each sort and obtaining the transitive closure of the union (Hernandez and Stolfo, 1995, 1998).

There is also an incremental method to cleaning databases based on this multiple-sort sorted neighbourhood algorithm. In this algorithm, the objective is to not clean previously cleaned data, but to clean new data within the database as well as adjust the records about the previously cleaned data to reflect this new cleaning. In the incremental sorted neighbourhood algorithm, new data is compared against keys that are 'representative' of the database. These representative keys are obtained from the previous cleanings. In each of the previous cleanings, similar records are grouped together or clustered. Then, from these clusters, a key describing the common attributes from the cluster is developed. When performing a new cleaning, the dirty records are compared against the representatives through the multi-pass version of the sorted neighbourhood method. If duplicates are detected during this phase, they are then merged or eliminated. Once the various sorts finish, based on the knowledge gained from the sorts, the record is placed into the most similar cluster. The representative is then modified to reflect the new record (Hernandez and Stolfo, 1995, 1998).

2.2 Knowledge-based methods

Besides using methods with various applications of sorting, there are a number of machine learning methods for performing data cleaning (Cochinwala et al., 2001; Low et al., 2001). One popular method is to incorporate knowledge bases into data cleaning tools. Knowledge bases provide domain dependent information that can be used to improve error detection techniques and detect duplicates. A good example of a knowledge base data cleaning tool is Intelliclean by Low et al. (2001). Intelliclean first 'scrubs' the data for irregularities that can be detected easily. For example, it standardises abbreviations. Therefore, if one record abbreviates the word street as 'St.' and another abbreviates it as 'Str.' while another record uses the full word, all three records can be standardised to the same abbreviation. Once the data has been scrubbed, it is then cleaned using a set of domain-specific rules that work with a knowledge base. These rules detect duplicates, merge appropriate records and create various alerts for any other anomalies. Finally, the cleaning is then validated through user interaction and inspection (Low et al., 2001).

While many data cleaning tools are automated, there are also other tools that require human expert interaction. The tool Potter's Wheel, developed by Raman and Hellerstein (2001) at the University of Berkeley in California, is an example of such a system. With interactive tools, an expert can use these tools to clean the data, exploiting the subtly of the expert's needs. Potter's Wheel is a domain independent tool. The interface allows the user to view the various cleaning rules being employed on the data set. From this view, the user can then modify the rules so that cleaning becomes more precise. The actual operation to clean the data is then done without the user needing to interact with the system (Raman and Hellerstein, 2001).

Another method, proposed by Caruso et al. (2000) uses data reconciliation techniques to match and combine duplicate records within a give database. In the tool they created, the use an extensible platform that uses machine-learning techniques to decide how to eliminate duplicate entries within a database. The tool uses a training set of data from a database it will eventually be applied to. From this training set, the tool can develop a set of rules for matching data to find duplications. Once the training set is selected, preprocessing of the data occurs. This pre-processing can include removing common words within the data set or reducing white space. Next, a set of measures of similarity is selected for the data. The tool is the trained on the selected data to obtain the rules for detecting duplication. It is then applied to the entire data set (Caruso et al., 2000).

2.3 ETL method

Currently, the most popular method for data cleaning is the ETL method. The ETL method performs data cleaning through an extraction, translation and loading process. During this process, two types of cleaning occur: on the instance-level and on the schema-level. Instance-level cleaning refers to errors within the data itself, such as misspellings. Schema-level cleaning usually concerns integrating the database into a new schema, a data warehouse or a federated database. In the ETL process, the data and schema are extracted, various operations are performed on the data and schema to clean them, and then the new schema in put into the database with the cleaned data (Rahm and Do, 2000). ETL's primary tools are data flow graphs, which tracks the transformation of the dirty data into the cleaned data (Rahm and Do, 2000).

The ARKTOS system, a data cleaning system that implements the ETL structure is an example of a system that uses this structure (Vassiliadis et al., 2001).

While each of aforementioned methods is interesting, they do have their drawbacks. The ETL method tends to be database-specific. The tools designed to perform the extraction, translation and loading tend to apply to only one database. The tool proposed by Caruso et al. (2000) only considers the need for matching and deleting duplicates. Also, the pre-processing phase where a set of data is selected for training has some disadvantages. First, it requires prior knowledge of what instances in the database might be problematic. Second, in the preprocessing field, the data also needs to be standardised before the learning algorithms are applied. The method that will be used for this project is the declarative method. In this method, various operators for data cleaning is conceptually defined. Then, tools are developed to implement these operators, depending upon the definition of the concept with respect to the data. For example, a match in a dictionary would be defined differently then a match for a protein in a biological database. This method has been implemented into the tool AJAX (Galahardas et al., 2001a, 2001b).

2.4 Disambiguation methods

Recently, many data cleaning efforts have learned toward methods that center on disambiguation. Popular fuzzy methods include the method created by Chauhuri et al. (2003) for approximate string matching to generate an Error Tolerant Index (ETI). Kalashnikov and Mehrotra have introduced a data-independent method for reference disambiguation using entity-relationship graphs (Kalashnikov and Mehrotra, 2006). Fuxman et al. (1978) have introduced "ConQuer, a tool which rewrites SQL queries to help resolve data inconsistencies that can occur while ordinary sequel queries". Chu et al. (2005) use belief propagation with Hidden Markov Models to disambiguate data.

Table 1 Table of popular data cleaning methodologies

Methodology	*Example system*
ETL	Artos (Vassiliadis et al., 2001)
Extensible machine learning techniques	Telcordia's database reconciliation and data quality analysis tool (Caruso et al., 2000)
Declarative model	AJAX (Galahardas et al., 2001a, 2001b)
Knowledge-base technique	Intelliclean (Low et al., 2001)
Duplicate detection through sorting	Bitton's pre-sorting (Bitton and DeWitt, 1983)
Multi-pass sorted neighbourhood	Merge/Purge (Hernandez and Stolfo, 1995, 1998)
User interaction	Potter's Wheel (Raman and Hellerstein, 2001)
Disambiguation methods	ConQuer (Fuxman et al., 2005)

2.5 Declarative data cleaning

Declarative data cleaning, as described by Galahardas et al. (2001a, 2001b) involves two processes: logical processes to perform data transformations on dirty data and performance processes to improve these transformations without sacrificing accuracy. This approach necessitates the creation of five conceptual operators which interact with

each other to perform data cleaning. These operators are as follows: MAP, VIEW, MATCH, CLUSTER and MERGE. The mapping operator would specify how to transform one relation into a second relation (Galahardas et al., 2001a, 2001b). This operator allows for schema transformations and schema integrations. The view operation is similar to an SQL query with modifications that can help with exception handling and integrity constraints (Galahardas et al., 2001a, 2001b). The matching operator computes the difference between two input tuples to determine the degree of similarity between the two tuples. This function can be used to detect duplicates within the database as well as detect errors against a knowledge base. Also, it can be modified so that degree of similarity can be computed. The cluster operation takes a relation that defines a set of elements from the database as input and returns one relation as output. This operation allows for the definition of that output relation to be a description of some distance within the initial set of elements. The merge operation defines a method for combining similar tuples into a defining relation. In merging, a single relation defining a set of elements is taken as input and one relation is returned as output. This relation groups and collapse the set of elements within the input relation based on some matching criteria (Galahardas et al., 2001a, 2001b). Based on this framework, data cleaning systems can be built that optimally address cleaning problems from a given data set.

Galahardas extended this framework into the tool AJAX. AJAX is a data cleaning system for publication and citation information. The cleaning operations are applied to this data, which is in a relational database format (Galahardas et al., 2001a, 2001b). They operations present an approach to perform various data cleaning on most data sets. Since they are not instantiated precisely, but rather can be instantiated into whatever data cleaning system a group wants to develop, they can be instantiated based on the data set. This preserves the concepts of the operations needed to perform data cleaning while also allowing flexibility in the way the specific data set needs from a particular operation. Therefore, these data cleaning operations can be applied to biological and evolutionary data. Moreover, the operations can also be applied regardless of whatever type of data model or schema is used to create the database (Galahardas et al., 2001a, 2001b).

3 BIO-AJAX: a case study

BIO-AJAX (Herbert et al., 2004) is a proposed data cleaning framework for biological databases that is designed to clean both schema level and data level data quality problems based on a declarative data cleaning framework. It uses the conceptual operations presented by Galahardas et al. (2001a, 2001b) that ultimately developed the AJAX system as well as adding a new operation to process outlier data from clustering. BIO-AJAX preserves conceptually the operations while modifying the cleaning operations so that they specifically apply to biological databases needs. These cleaning operations will help to reduce a number of the problems mentioned previously that are inherent to biological databases.

To illustrate BIO-AJAX best, phylogenetic data, specifically phylogenetic trees were chosen. There are a number of reasons for this choice. First, phylogenetic data tends to be a heterogeneous dataset. The main components of this data set are structures in the form of phylogenetic trees modelled in various ways. For a standard keyword database, the problem of data quality and data cleaning has been well explored. However, the problem of cleaning databases containing structural data has not been researched. Moreover, most

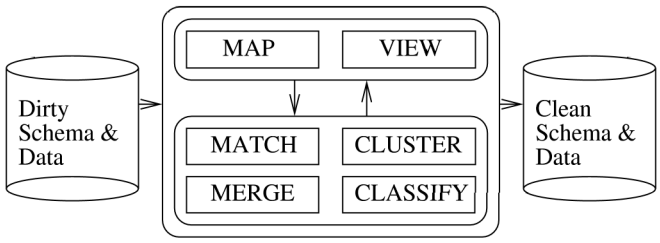
phylogenetic databases have other data associated with it. Therefore, this also motivates the problem of how does a data cleaning tool operate upon databases with different data. With heterogeneous data come the issues concerning how does the tool measure similarity as well as cluster or merge that data. With heterogeneous data come the issues concerning how does the tool measure similarity as well as cluster or merge that data. Therefore, phylogenetic data can demonstrate both of these problems as well as provide an interesting data set to perform cleaning on.

To clean biological data, some modifications are needed of the aforementioned framework. While the framework is powerful for many data cleaning problems, there are some extensions that biological data, particular phylogenetic data need so that the data is fully cleaned. For phylogenetic data, each of the five operators would need to be extended specifically for the data set. Moreover, there can be multiple extensions of these operators. For example, there are a number of algorithms that can perform matching in any data set, let alone phylogenetic data. Also, for the framework, a sixth operator has been added. This operator, CLASSIFY, will perform classifications within the database and clean it accordingly.

3.1 The MAP and VIEW operators

The mapping operation can help with a number of problems inherent within these databases. Mapping is key to both biological schema cleaning and biological data cleaning. First, the mapping operation can help with transferring legacy data into new schemas. Second, it can help populate the database with data from other databases by transforming the second database's data schema into the first database's schema. Finally, if there is a schema level problem, mapping can map the data in the dirty schema into a better schema. The view operation should be standard with respect to what has already been said since it is a method for querying the database. For example, the view operator can be used to display the tables generated with mapping operator.

Figure 2 The BIO-AJAX framework



3.2 Mapping and viewing phylogenetic trees

One common problem with phylogenetic trees within databases is that the trees can be stored in various formats. Therefore, one possible purpose for mapping a data set of trees is to map them into the same format. If the trees are in different formats, a number of problems can occur. First, to perform any knowledge discovery or even comparison operations on the data set, ideally the data should be in the same format. If not, any tool written to process the data will need to be able to process all formats of the data. However, this method does not manage possible future cases. If the database modifies the

format even slightly, any new data will be lost. Mapping the format of the trees helps to accomplish a number of data cleaning objectives. First, it allows for legacy formats within the database to be standardised. This will then allow for all data to be available for any knowledge discovery tool written for that specific format. Second, any data that is not in a needed specific format can then be mapped onto that format. Third, if phylogenetic databases exchange data, mapping can help integrate new data into the pervasive format within the database. Finally, if a database needs to change or update to a new format, the legacy data can be easily updated as well. The view operation should be standard with what has already been said since it is a method for querying the database. For example, the view operator can be used to display the tables generated with mapping operator.

3.3 The MATCH operator

The matching operator has many purposes that can help facilitate data cleaning. First and foremost, it is an excellent tool for detecting duplicates or records whose semantic contents are extremely similar, that in essence the records are duplicates. For many of the biological fields that have readily accessible databases, exact matching and similarity matching are well-explored research areas. For example, for a database containing nucleotide or amino acid sequences, sequence alignment algorithms such as BLAST, FASTA and CLUSTAL-W can tell a user how similar two sequences or a set of sequences are. Moreover, while these areas are well explored, most of these areas so do not have similarity measures that can be considered perfect for every comparison.

However, a large problem within biological research is the idea of detecting synonymous data. Since biological data can be complex, incorporating structures and well as text, detecting entries that are not identical syntactically but are identical semantically is a large problem. For example, it is common for the more heavily research species to have multiple identification names. Most species at least have their scientific name and their language-specific vernacular name (e.g., Human and homo sapien, fruit fly and drosophila melanogaster). This problem can be compounded by a number of factors. First, the scientific or Linnaean names for organisms can be inadequate for describing a species. Many species were classified before genetic sequencing and other more quantitative methods were developed to classify a species. With the use of the more qualitative methods, it has been the case where that separately classified species ultimately were variations within the same species. Since many phylogenetic databases have legacy data, and there are many sources for phylogenetic information, both standardising nomenclature, or the naming mechanisms used to identify a species throughout a set of data, through trees as well as checking nomenclature within a knowledge base can help clean the data significantly. By standardising nomenclature, comparisons and other knowledge extraction tasks can be performed much more effectively and efficiently.

Besides nomenclature, matching can also use used for error detection. In biological data, there are a number of causes for errors. Like any other database, biological databases can easily suffer the same problems of improper input, spelling mistakes and non-standard abbreviations. However, if used with a knowledge base, matching can be a very effective method for detecting errors (Lacroix and Critchlow, 2003).

A knowledge base for species can be preliminarily developed from readily available web databases. For example, NCBI provides a taxonomy tool that could be exploited into

a knowledge base. The NCBI taxonomy tool, located at http://www.ncbi.nlm.nih.gov/Taxonomy/taxonomyhome.html/ gives a great amount of information about any given species. By querying these tools with a given species name, the various nomenclature about the species can be extracted from the resulting web pages. Once the nomenclature is extracted from the Web pages, it can be used to standardise the nomenclature in the phylogenetic trees.

Also, duplicate or similarity detection is very important in phylogenetic studies. When a duplicate is detected, this could mean that one of many different events have occurred within the database. First, there could be a duplicate record within the database. In that case, the record should be removed. Another case may be that there exists two identical trees by the same author is two different studies.

The curator of the database may want to decide then how to handle this duplicate information. Next, there may exist duplicate trees within the database by different author using different reconstruction algorithms. This could be a potentially important find since most algorithms that create phylogenetic trees do not perform the exact same operations as any other reconstruction algorithm. This information could also be helpful to the users of the database, if they want to see similar trees. The matching operator can incorporate a number of matching algorithms that are native to either biological data sets or specifically phylogenetic tree data sets. Also, there are also a set of matching algorithms, while not native to biological databases, are customarily used on keyword databases that can be used to process phylogenetic trees, especially the metadata associated with the trees. However, these algorithms must be used so that a match can be detected on the biological or phylogenetic data. For example, the matching operator can include sequence alignment, structure comparison as well as a thesaurus to handle the species synonymy problem. With the various terminology as well as the different methods for analysing the structures involved within biological data sets, it is very possible to have duplicate records, based on a given biological measure that is not evident through keyword comparison.

3.4 The CLUSTER operation

The clustering operation can perform similar duplicate detection functionalities that the matching operator can perform. Conceptually, the cluster operation organises a set of elements in a relation by either their value or their distance from one another. Both ‘value’ and ‘distance’ can be defined for any given database set, especially biological data, so that this operation can be performed. For example, for protein data, this can be performed for proteins that have similar structures or amino acid sequences. For nucleic databases, clustering can occur on sequences that have a specific gene or protein or segment of DNA that is similar. For phylogenetic data, this can be performed on the set of phylogenetic trees that have a similar structure.

Clustering phylogenetic data has been an important function within the field for a very long time. By clustering phylogenetic data, relationships between species can be discovered. Also, through clustering phylogenetic trees, comparisons of various reconstruction algorithms can be made. By clustering phylogenetic trees specifically, methods for creating the trees, or reconstruction algorithms, can be compared. Clustering allows for scientists as well as the database manager to see what data can be considered similar and what is possibly an aberration within the database. This can help with error detection since clustering can highlight outliers. Outliers can then be analysed and

cleaned appropriately. Moreover, through these comparisons, various identifying aspects of the trees can be learned. For example, by clustering similar trees, scientists can learn how closely species or groups of species are related to each other. Reconstruction algorithms can also be evaluated. It can demonstrate the effectiveness of an algorithm as well as its flaws. For example, give a given set of peer-reviewed trees, clustering can find the commonality structure between two trees. Moreover, if a user would like to compare his or her tree against similar trees, clustering can help the user see what trees his or her tree should be compared against. If the tree is then compared to the database, and the new tree is not included within the cluster, this could indicate a problem with the reconstruction algorithm or tool that created the tree (Stockham et al., 2002).

Clustering phylogenetic trees is a complicated topic. Currently, the most popular approaches use a combination of popular clustering methods and consensus trees. In these phylogenetic tree clustering algorithms, standard clustering algorithms such as K-means and agglomerative clustering can be applied to the phylogenetic trees. Then, once the trees are in clusters, consensus trees can be formed from these clusters to represent the clusters. A consensus tree is a tree T where, given a set S of trees, all edges in T are contained within every member of S. The formation of a consensus tree creates a representative for a clustering of phylogenetic trees. Consensus trees can be used without first clustering through the standard algorithms, however this creates one tree for the entire data set (Stockham et al., 2002).

3.5 The outlier operator as a part of the CLUSTER operator

While the aforementioned operators, it is possible to detect and correct a number of instances where data is considered ‘dirty’. However, errors within the data can still pass unnoticed. Moreover, with clustering and merging operators, it becomes important to analyse the results of the clustering operation in a more in-depth manner. While clustering can help detect possibly similar or duplicate records, outlier records can also indicate errors in the data.

An outlier is a data point that is different than the rest of the data within a database for some given measure. Outliers can be detected easily within a data set that has been clusters, since the data will not be contained in any cluster. For the data point not to belong to any cluster, this means that the data this specific data point represents is not similar to any other data within the clustering based on.

Concerning phylogenetic data, outlier detection can yield a great amount of information. First, concerning data cleaning, outlier detection can indicate a problem with the data. If a given tree T created with a set of taxon S and a reconstruction algorithm R does not behave in an expected manner, such as fall into clusters with other trees mapping S with R, then there could be an error with the data. This error can be the result of a number of problems. Some of these possible errors are: input errors by the creators, improper instantiation of the reconstruction algorithm, improper application of a reconstruction algorithm, and faulty data used to create the tree.

3.6 The MERGE operator

The merge operation can act similarly to match and cluster. Since merging occurs based on ‘value’ or ‘distance’ with respect to a given attribute, the operations that can be performed are very similar to those in clustering. However, since the data is grouped and

collapsed, this might be a useful method for creating consensus trees, supertrees or superstrings. Also, if duplicates are detected within the match phase, merge can be used to reduce a set of elements that are identical into one element.

Merging phylogenetic trees is commonly done in various application of phylogenetic research. Usually trees are merged through the use of supertrees (Sanderson et al., 1998). Supertrees are a pervasive method throughout the study of phylogenetics for relating phylogenetic subtrees to each other. The purpose of the supertree is to take a set of phylogenetic subtrees, which may or may not contain the some of the same species, and for one tree. This one tree preserves as best as possible the evolutionary relationships between the species within all of the trees (Sanderson et al., 1998; Sharkey and Leathers, 2001).

Since scientists studying phylogenetics theorise life originated with once species, and all other species evolved from that one, the super tree models a possible pattern for this evolution. It specifies the relationship between speciess, and with supertrees, it also specifies the relationship between trees. While supertrees can be used to merge any set of trees, it offers an interesting capacity for phylogenetic data cleaning in that it can act as a merge function for phylogenetic data. If two tree records are detected to be similar enough for merging, the supertree algorithms can be used to merge the trees within the phylogenetic records (Sanderson et al., 1998; Sharkey and Leathers, 2001).

3.7 The CLASSIFY operator

The final operator for the BIO-AJAX framework is the CLASSIFY operator. In working with biological data, performing classifications is a routine procedure. Classification aids biologists and database curators alike. First, classification offers the ability to discover patterns within the data if no known patterns already exist. Moreover, it also allows us to explore known patterns deeper as well as possibly discovering more patterns. Next, classification also helps with the data item has any of a variety of data quality issues concerning it.

Due to the nature of the biological data, especially due to the enormous size and complexity of it, pattern recognition within the data set becomes extremely important. Through classifying the data, previous observations can be confirmed and enriched. First, if the new data item fits the classification model within given parameters, it adds credence to the classification model. Moreover, if it does not fit the model, it can indicate that there is an error either in the data or in the classification model. By adding to the knowledge of the patterns within a given data set, it helps biologists to better understand the underlying mechanisms within biology as well as model the data better with respect to these mechanisms.

Also, classification can help with a number of data quality issues. For example, within protein data, often a protein will be researched without fully understanding its function. Through classifying the protein, the protein that is most similar to it can be found. If the proteins are very similar, some aspects of the information stored about the known protein may be also connected to this unknown protein, helping to improve the consistency of the data. Moreover, it helps to disseminate metadata, which is currently a key concern for all biological databases.

Concerning phylogenetic data, there are many instances where classification may be needed. Concerning nomenclature problems, classification can offer the ability to standardise a set of nomenclature to a specific format by classifying whether or not it already exists in a certain format.

Also, classification can help in analysing phylogenetic tree structures. If a reconstruction method is unknown or if there is inconsistent data, classification can aid in improving these problems. It can also identify characteristics of reconstructions as well as any abnormalities within a construction.

4 Implementation and experimental results

BIO-AJAX is implemented using Perl, JAVA, JSP, HTML, and placed over TreeBASE as middleware to preserve data integrity. Figure 3 shows the interface of the system. Due to the sensitive nature of biological data, specifically in this case phylogenetic trees, BIO-AJAX has been designed to never alter original submission data, but rather to interact with the original data and provide facilities to clean the interactions with the data. This is necessary, especially in an archival biological database cleaning since the data is associated with publications by the researchers.

To perform the nomenclature cleaning, first, all of the taxa are extracted from TreeBASE. The taxa are extracted with the name of the experimental study file that contains them. Once the taxa are extracted, they are organised lexicographically according to taxon names. The extraction file then goes through a rudimentary cleaning phase. This cleaning phase removes characters that could not possibly be a part of the taxon names as well as formats the extraction file for interaction. These characters, such as a forward slash before a taxon name, were determined to be extraneous characters during the analysis phase in the BIO-AJAX implementation for TreeBASE. Next, the file is formatted to become input for a prefix generation tool.

The prefixes are generated by producing all possible prefixes containing the first word of the nomenclature. (Most nomenclature consists of more than one word and the first word is an identifying term of a species.) For example, given the taxon ‘Homo sapiens x’, the following prefixes would be generated: ‘Homo’, ‘Homo s’, ‘Homo sa’, ‘Homo sap’, ‘Homo sapi’, ‘Homo sapie’, ‘Homo sapien’, ‘Homo sapiens’, ‘Homo sapiens x’. If the taxon name is one word long, then the prefix containing only that one word is created for that taxon ensuring every taxon in TreeBASE will be tested. Once the prefix list has been created, this list is then automatically used as input for the NCBI TaxBrowser query tool (Benson et al., 2000; Federhen et al., 2004).

The NCBI Taxonomy database (Benson et al., 2000; Federhen et al., 2004), considered as an excellent resource within the field, is a repository of phylogenetic and taxonomic data about various species. The NCBI Taxonomy database provides tools to search and browse phylogenetic data about most species. This method can be implemented upon any taxonomy reference, with modifications made to the wrapping mechanisms for that repositorys data. BIO-AJAX for TreeBASE conglomerates data from multiple resources about taxa and species. This data is dynamically updated as the Taxonomy database is updated, representing a concise representation of peer reviewed phylogenetic data. Therefore, it provides an ideal resource for solving the nomenclature problem in TreeBASE. Due to the storage nature of phylogenetic databases, the wrappers used to extract the data are database-dependent.

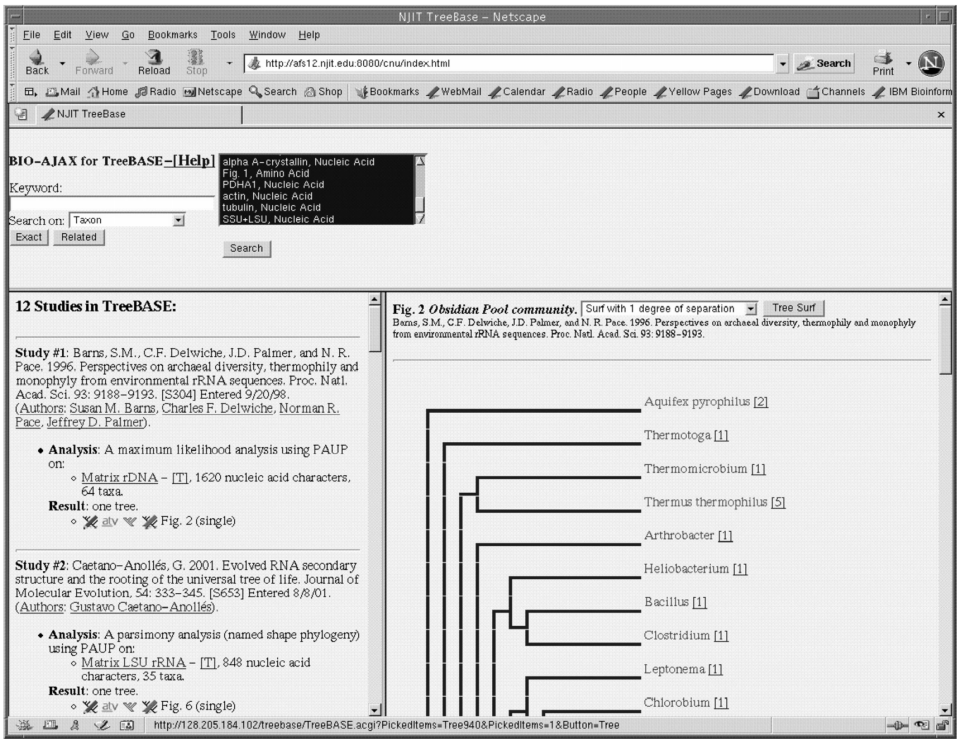
The list of prefixes is queried against the NCBI Taxonomy database's search tool TaxBrowser. This tool offers the user a number of options for searching for taxa. For BIO-AJAX's purposes, the following four types of searches are used: Complete Name search, Wild Card search, Token Set search and Phonetic Name search. The search for each prefix yields one of three possible results. First, the prefix is exactly found within the NCBI taxonomy database. If this is the case, the tool has found a 'data page' about the prefix. The prefix and all other nomenclature NCBI associates with that entry are indexed. The results from each of these searches are then combined together. For example, consider again T_1 in Figure 2 where 'Homo sapiens' is a prefix of 'Homo sapiens x'. When sending this prefix to the NCBI taxonomy database, we will get 'human', 'man' and 'Homo sapiens' returned, and therefore all the three taxon names are linked with T_1. Similarly, when sending the prefix 'human' ('man', respectively) of 'human A' ('man 1', respectively) in T_2 (T_3, respectively) to the NCBI taxonomy database, we will also get 'human', 'man', and 'Homo sapiens' returned. Thus, the three taxon names will be linked to all the three trees T_1, T_2 and T_3.

In the second case where the prefix does not return any match, then the prefix is discarded. Finally, if the prefix returns a hierarchical listing of possible matches, then an exact phrase search is performed within the list. If the exact prefix is found in the list, then that link is explored and treated as a data page. Only the exact match is used since, if each result in the list were used, then many taxa that are not related significantly enough to the original TreeBASE taxon would be included. For example, if the original TreeBASE taxon is 'Homo sapiens', the prefix 'Homo' would be generated from that taxon. The query 'Homo' on TaxBrowser results in a list containing the taxa 'Homo', 'Homo sapiens', and 'Homo sapiens neanderthalensis'. Only the taxon 'Homo' will be explored. 'Homo' represents a genus that Homo sapiens belong to. Therefore, data about this group may be of interest to a user. If all of the list were explored, 'Homo sapiens neanderthalensis' would have also been included as a possible association to 'Homo sapiens'. Since these are two distinct species, this is a relationship that should be eliminated when creating the associations.

Once the list is obtained from NCBI, it is indexed using hash tables. These tables allow for the exploitation of both the original data from TreeBASE and the data obtained from the prefix generation and querying. One index is comprised of the original taxa obtained from TreeBASE. The other index is comprised of the prefixes and other nomenclature obtained during the NCBI Taxonomy verification stage.

Now consider again the BIO-AJAX interface shown in Figure 3. This interface has been modelled similarly to the TreeBASE interface. The user can enter a nomenclature query in the top frame. This query is then formatted for interaction with the index. The query is checked against both indices described above. If a match is found in the index reflecting the original TreeBASE taxa, this match is considered an 'Exact' match and any data linked to this match is highlighted as an exact match. If a match is found in the index that contains the nomenclature obtained from NCBI and the prefixes, then these matches are treated as 'Related' matches. With each index, the matrix accession for its related studies is also housed with it. Once the matches are found, the data is then extracted from the TreeBASE database through using the matrix accession numbers of the studies. The results are then formatted, with the exact matches listed first, and displayed to the user similarly as to how the results are displayed on the TreeBASE website.

Figure 3 The BIO-AJAX interface implemented for TreeBASE



When searching, if the user wants to search for the taxon with exact matches in TreeBASE, he or she clicks on the 'Exact' button. If the user wishes to search for any related matches to the taxon, he or she would click on the button 'Related'. The results of the search appear in the text box in the top frame. From there, the user can select one to all studies to be displayed. Once the studies are selected, they can be displayed in the bottom left frame. Data from these studies, including the phylogenetic trees, can be displayed in the bottom right frame.

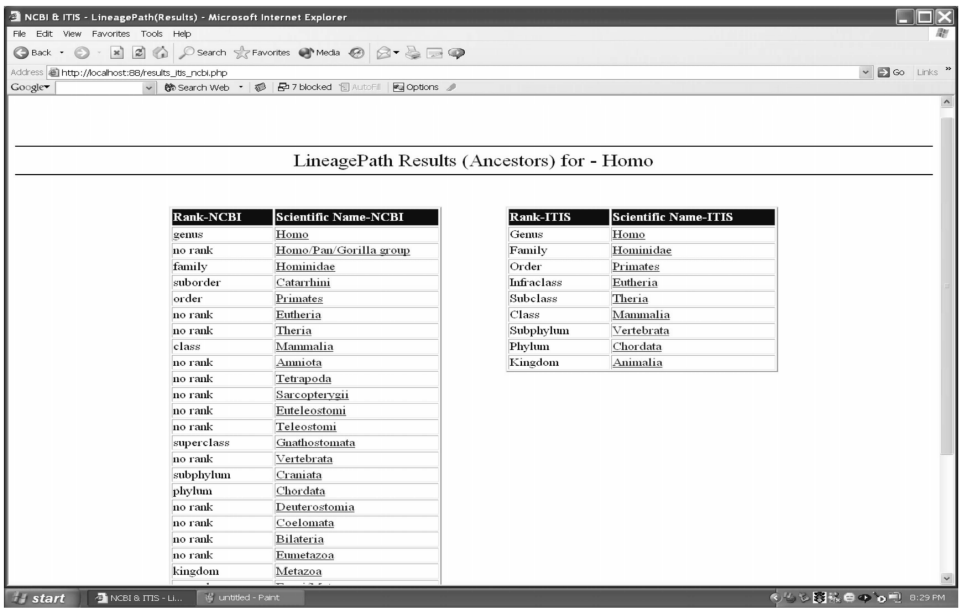
The results in Table 2 reflected how BIO-AJAX found a number of taxa related to the TreeBASE taxa that were not previously detected. This has a number of implications for the cleaning method as well as TreeBASE. First, within the studies archived in TreeBASE, while efforts have been made to minimise the nomenclature problems, only approximately 50% of the taxa in TreeBASE are recognisable by the standards within the phylogenetic community stored in the NCBI Taxonomy database. Moreover, since many data repositories link to TreeBASE, such as the NCBI Taxonomy database, this has implications for them as well. Second, with the prefix generation technique, the recognition of the taxa can be improved to approximately 88%. This helps to solve the inconsistency and incompleteness problems concerning the nomenclature in TreeBASE as well as any other tool that links to TreeBASE.

Table 2 Results of taxa searches in NCBI TaxBrowser with TreeBASE nomenclature

Search option	*TreeBASE taxa*	*With prefixes*
Complete name	13,076 (49%)	23,493 (88%)
Wild card	13,079 (49%)	23,493 (88%)
Token set	13,172 (49%)	23,801 (89%)
Phonetic name	14,025 (52%)	24,633 (92%)

The current implementation for BIO-AJAX for Lineage Paths (Herbert et al., 2005) uses the data warehousing technique for integrating the data and is accessible through a World Wide Web interface (Hernandez and Kambhampati, 2004). Figure 4 displays the interface for this tool. The lineage paths are extracted from both NCBI Taxonomy Database and ITIS and stored locally. In previous version of the tool and previous instantiations of BIO-AJAX, the mediator method was used to display the paths. However, due to limitations in accessing each repository, this method had to be replaced with the warehousing method. Moreover, since each lineage path needed to be manipulated to get very specific data out of it, the mediator method became impractical. For comparison purposes, the mediator method performed an adequate job. However, for finding all ancestors and descendents, the lineage path strings needed to be parsed creating a long lag time for the Web interface. Therefore, the platform was shifted from using the mediator method to the data warehousing method.

Figure 4 The BIO-AJAX interface for showing an example output for an ancestor query for the taxon 'Homo'



5 Conclusion

After discussing the possible operations that can be performed through data cleaning on biological database, there are a number of effects on biological information retrieval and data mining. Primarily, since the data has been processed to cater to the needs of a particular database, the data is now in a standardised format for whatever processing the users need. Since the data, after cleaning, has been organised and translated into a concise format, many of the tools for information retrieval and data mining can perform better.

First, the data will necessarily be more organised from the cleaning operations. Through the mapping, matching and merging operations, data will be transformed into the schema that database can analyse most effectively. Moreover, hard to detect duplicates should be merged into one entry. Also, errors are detected and dealt with before a user can access the database. Since the occurrence of duplicates has been minimised and the data has been transformed into a preferred schema helping to eliminate errors, information retrieval tools can work more efficiently. The database management system should perform better with respect to recall and precision, there should be less need for complicated indexing algorithms such as stemming and, through the clustering and match operations, similarity should be easier to detect.

Data cleaning also gives data mining algorithms more precise and standardised data. With this standard data, common data mining techniques will run more effectively. If there are errors within the data or if the data in not organised properly, data mining algorithms can see these errors as interesting aberrations within the data rather than error (Dasu and Johnson, 2002, 2003). Also, each of the operations within this declarative data cleaning model can be seen to perform data mining tasks. Mapping translates one schema into another schema. Matching, clustering and merging each act upon information concerning the degree of similarity a set of data has with respect to a particular measure. From this, especially from clustering and merging, patterns can be obtained. Also, since data cleaning can eliminate errors and duplicates, this helps to streamline the data. This helps with the compression issues involved with data streams and data squashing (Dasu and Johnson, 2002, 2003).

This framework offers many possibilities for future research in many different areas. Possible research includes explaining the framework, specifying the algorithms that govern the framework more precisely, exploring improvements on already existing algorithms for the current framework and applying the framework to other biological data besides phylogenetic data.

First, the framework needs to be further specified to be instantiated upon Tree-BASE. This paper explains some possible operations that can be performed on the phylogenetic data contained within the database TreeBASE. It cites some possible algorithms and methods for performing the operations of BIO-AJAX. However, ultimately, only a few of the operations can be performed to maintain consistency throughout the database. For example, it might be beneficial to implement only one algorithm for matching trees. Further research can include instantiating this tool to operate specifically upon TreeBASE, working with its curators to find the various measures needed to perform the aforementioned operations.

Next, while matching phylogenetic trees is a well-explored area, clustering, merging, outlier detection and mapping are not. While there are algorithms that are available to perform these functions in a simple manner, there are no algorithms that handle the more pervasive complex methods needed to fully implement the generally accepted ideas of

clustering, merging, outlier detection and mapping. Each of the operators, even with the simple algorithms, also offers a great opportunity for improvement. For matching, developing a knowledge base or interacting with previously established knowledge bases for the sole purpose of nomenclature standardisation is not well research for phylogenetic databases. This problem has been addressed for species concerning protein databases and nucleotide databases. However, these databases usually do not act as a repository for phylogenetic trees.

Also, by solving these problems, a number of curator and user interaction problems develop. Issues concerning how involved the curator should be during the cleaning process, especially concerning duplicate diction, elimination and merging. Also, within the matching, clustering, merging and outlier operators, it was mentioned how the results of those operations could help the user discover more knowledge about the data. How to convey what cluster a particular tree is in or what trees it was merged with to the user becomes an issue.

Data integration is key to helping biological databases facilitate the knowledge discovery processes within biological research. Through integration, biological databases are afforded the tools to solve some of their large problems. It allows these repositories the freedom to integrate new data or data not held within their repository into a unified view for a user. It also gives the curators the ability to use multiple database management systems to help describe their data more completely. Moreover, integration allows the user the freedom of querying in a unified interface, rather than needing to visit multiple databases and then collate the results. In this paper we presented three areas of biological databases that can be aided with integration technologies. By applying integration techniques to these areas, some difficult problems facing biological databases can be solved.

Finally, when this framework was developed, it was intended to be independent of a specific database. Therefore, it would be interesting to apply this framework to possibly other biological databases such as a nucleotide database or a protein database.

Acknowledgement

We would like to thank Helen Berman, Narain Gehani, Roderic Page, William H. Piel, Shashikanth Puspati, Dennis Shasha, John Westbrook and Cathy Wu for useful discussions concerning this work.

References

Benson, D.A., Karsch-Mizrachi, I., Lipman, D.J., Ostell, J., Rapp, B.A. and Wheeler, D.L. (2000) 'GenBank', *Nuc. Acids Res*., Vol. 28, No. 1, pp.15–18.

Bitton, D. and DeWitt, D.J. (1983) 'Duplicate record elimination in large data files', *ACM Transactions on Databases*, Vol. 8, No. 2, pp.255–265.

Caruso, F., Cochinwala, M., Ganapathy, U., Lalk, G. and Missier, P. (2000) 'Telcordia's database reconciliation and data quality analysis tool', *Proc. 26th International Conference on Very Large Data Bases*, Cairo, Egypt, pp.615–618.

Chaudhuri, S., Ganjam, K., Ganti, V. and Motwani, R. (2003) 'Robust and efficient fuzzy match for online data cleaning', *Proc. 2003 ACM SIGMOD International Conference on Management of Data*, San Diego, California, USA, pp.313–324.

Chu, F., Wang, Y., Parker, D.S. and Zaniolo, C. (2005) 'Data cleaning using belief propagation', *Proc. Second International ACM SIGMOD Workshop on Information Quality in Information Systems*, Baltimore, Maryland, USA, pp.99–104.

Cochinwala, M., Kurien, V., Lalk, G. and Shasha, D. (2001) 'Efficient data reconciliation', *Information Sciences*, Vol. 137, Nos. 1–4, pp.1–15.

Dasu, T. and Johnson, T. (2002) 'Problems, solutions and research in data quality', *Proc. SIAM International Conference on Data Mining*, Arlington, Virginia, April 11–13.

Dasu, T. and Johnson, T. (2003) *Exploratory Data Mining and Data Cleaning*, John Wiley & Sons, Hoboken, New Jersey.

Federhen, S., Harrison, I., Hotton, C., Leipe, D., Soussov, V., Sternberg, R. and Turner, S. (2004) *NCBI Taxonomy Homepage*, http://www.ncbi.nlm.nih.gov/Taxonomy/taxonomyhome.html/.

Fuxman, A., Fazil, E. and Miller, R.J. (2005) 'ConQuer: efficient management of inconsistent databases', *Proc. 2005 ACM SIGMOD International Conference on Management of Data*, Baltimore, Maryland, USA, pp.155–166.

Galahardas, H., Florescu, D., Shasha, D., Simon, E. and Saita, C.A. (2001a) 'Declarative data cleaning: language, model and algorithms', *Proc. 27th International Conference on Very Large Data Bases*, Rome, Italy, pp.371–380.

Galahardas, H., Florescu, D., Shasha, D., Simon, E. and Saita, C.A. (2001b) *Declarative Data Cleaning: Language, Model and Algorithms*, INRIA Technical Report RR-4149.

Greer, D.S., Bourne, P.E. and Berman, H.M. (2002) 'The protein data bank: unifying the archive', *Nucleic Acids Research*, Vol. 30, No. 1, pp.245–248.

Harlin, M. (2003) 'Taxon names as paradigms: the structure of nomenclatural revolutions', *Cladistics*, Vol. 19, pp.138–143.

Herbert, K.G., Gehani, N.R., Wang, J.T.L., Piel, W.H. and Wu, C.H. (2004) 'BIO-AJAX: an extensible framework for biological data cleaning', *ACM SIGMOD Record*, Vol. 33, No. 2, pp.51–57.

Herbert, K.G., Pusapati, S., Wang, J.T.L. and Piel, W.H. (2005) 'Lineage path integration for phylogenetic resources', *Proc. 17th International Conference on Scientific and Statistical Database Management*, Santa Barbara, California, USA, pp.117–120.

Hernandez, M.A. and Stolfo, S.J. (1995) 'The merge/purge problem for large databases', *Proc. 1995 ACM SIGMOD International Conference on Management of Data*, San Jose, California, USA, pp.127–138.

Hernandez, M.A. and Stolfo, S.J. (1998) 'Real-world data is dirty: data cleansing and the merge/purge problem', *Data Mining and Knowledge Discovery*, Vol. 2, No. 1, pp.9–37.

Hernandez, T. and Kambhampati, S. (2004) 'Integration of biological sources: current systems and challenges ahead', *ACM SIGMOD Record*, Vol. 33, No. 3, pp.51–60.

Kalashnikov, D.M. and Mehrotra, R. (2006) 'Domain-independent data cleaning via analysis of entity-relationship graph', *ACM Transactions on Database Systems*, Vol. 31, No. 2, pp.716–767.

Lacroix, Z. and Critchlow, T. (Eds.) (2003) *Bioinformatics: Managing Scientific Data*, Morgan Kaufmann, San Francisco, California.

Lenzerini, M. (2002) 'Data integration: a theoretical perspective', *Proc. 21st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, Madison, Wisconsin, pp.233–246.

Low, W.L., Lee, M.L. and Ling, T.W. (2001) 'A knowledge-based approach for duplicate elimination in data cleaning', *Information Systems*, Vol. 26, No. 8, pp.585–606.

Piel, W.H., Donoghue, M. and Sanderson, M.J. (2000) 'TreeBASE: a database of phylogenetic information', *Proc. the International Joint Workshop for Studies on Biodiversity*, Tsukuba, Japan.

Rahm, E. and Do, H.H. (2000) 'Data cleaning: problems and current approaches', *Bulletin of the Technical Committee on Data Engineering, Special Issue on Data Cleaning*, Vol. 23, No. 4, pp.3–13.

Raman, V. and Hellerstein, J.M. (2001) 'Potter's wheel: an interactive data cleaning system', *Proceedings of 27th International Conference on Very Large Data Bases*, Rome, Italy, pp.381–390.

Sanderson, M.J., Purvis, A. and Henze, C. (1998) 'Phylogenetic supertrees: assembling the trees of life', *Trends in Ecology and Evolution*, Vol. 13, No. 3, pp.105–109.

Sharkey, M.J. and Leathers, J.W. (2001) 'Majority does not rule: the trouble with majority-rule consensus trees', *Cladistics*, Vol. 17, No. 3, pp.282–284.

Soares, C. (2004) 'Whats in a name?', *Scientific American*, November.

Stockham, C., Wang, L.S. and Warnow, T. (2002) 'Statistically based postprocessing of phylogenetic analysis by clustering', *Bioinformatics Discovery Notes*, Vol. 1, No. 1, pp.1–9.

Vassiliadis, P., Vagena, Z., Skiadopoulos, S., Karayannidis, N. and Sellis, T.K. (2001) 'ARKTOS: towards the modeling, design, control and execution of ETL processes', *Information Systems*, Vol. 26, No. 8, pp.537–561.

Wang, R.Y., Ziad, M. and Lee, M.Y. (1995) *Data Quality*, Kluwer Academic Publishers, Boston, Massachusetts.