ABSTRACT

NEW TECHNIQUES FOR IMPROVING BIOLOGICAL DATA QUALITY THROUGH INFORMATION INTEGRATION

by
Katherine Grace Herbert

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 standardized, methods and frameworks must be developed to handle both structural and traditional data.

The BIO-AJAX framework has been developed for solving these problems through both data cleaning and data integration. This framework exploits declarative data cleaning and exploratory data mining to help improve biological data quality. Within the framework, the problems and difficulties for cleaning biological data are discussed, specifically concerning phylogenetic data. Subsequently, it is demonstrated how BIO-AJAX can be used to improve the quality of the data in phylogeny. Moreover, within the cleaning architecture, data integration plays a key role in improving data quality. Data integration, due to the distributed and heterogeneous nature of the data sets becomes an enabling technology for improving data quality.

From these concepts, two tools have been built, BIO-AJAX for TreeBASE and BIO-AJAX for Lineage Paths. Each system approaches the data quality problem using the conceptual BIO-AJAX framework while implementing different methods working within the framework to improve the data quality.

NEW TECHNIQUES FOR IMPROVING BIOLOGICAL DATA QUALITY THROUGH INFORMATION INTEGRATION

by
Katherine Grace Herbert

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology and
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy in Computer Science

Department of Computer Science

May 2004

APPROVAL PAGE

NEW TECHNIQUES FOR IMPROVING BIOLOGICAL DATA QUALITY THROUGH INFORMATION INTEGRATION

Katherine Grace Herbert

Dr. Jason T.L. Wang, Dissertation Advisor
Professor of Computer Science, NJIT
Date

Dr. Narain H. Gehani, Committee Member
Professor and Chairman of Computer Science, NJIT
Date

Dr. James A.M. McHugh, Committee Member
Professor of Computer Science, NJIT
Date

Dr. Frank Shih, Committee Member
Professor of Computer Science, NJIT
Date

Dr. Vincent Oria, Committee Member
Assistant Professor of Computer Science, NJIT
Date

Dr. Michael M. Yin, Committee Member
Senior Technical Staff Member, AT&T
Date

BIOGRAPHICAL SKETCH

Author:	Katherine Grace Herbert
Degree:	Doctor of Philosophy
Date:	May 2004

Undergraduate and Graduate Education:

- Doctor of Philosophy in Computer Science,
  New Jersey Institute of Technology, Newark, NJ, May 2004
- Master of Science in Computer Science,
  New Jersey Institute of Technology, Newark, NJ, January 2001
- Bachelors of Science in Mathematics and Computer Science
  Saint Peters College, Jersey City, NJ, May 1999

Major: Computer Science

Presentations and Publications:

Katherine G. Herbert, Narain H. Gehani, Jason T.L. Wang, William H. Piel, and Cathy H. Wu., "BIO-AJAX: An Extensible Framework for Biological Data Cleaning," *ACM SIGMOD Record: Special Issue on Data Management in Life Sciences* , vol. 33, no. 2, June 2004, to appear.

Katherine G. Herbert, Jason T.L. Wang and Jianghui Lui, "Information Retrieval and Data Mining," *The Computer Science and Engineering Handbook*, Second edition, Chapter 75, Allan B. Tucker, Ed., CRC Press, to appear.

Katherine G. Herbert, John Westbrook and Jason T.L. Wang, "Data Integration in Biological Databases," *the Proceedings of the 7th Joint Conference on Information Sciences: the Atlantic Symposium on Computational Biology and Genome Information Systems & Technology*, pp.895-898.

Sen Zhang, Jason T.L. Wang and Katherine G. Herbert, "XML Query by Example," *The International Journal of Computational Intelligence and Applications, Special Issue on Internet Intelligence Systems*, World Scientific Publishing, vol 2, no. 3 September 2002. pp 329-338.

Huiyuan Shan, Katherine G. Herbert, William Piel, Dennis Shasha and Jason T.L. Wang, "A Structure-Based Search Engine for Phylogenetic Databases," *The Proceedings pf the 14th International Conference on Scientific and Statistical Database Management*, pp 7 -10.

Jason T. L. Wang, Qicheng Ma and Katherine G. Herbert, "Software Engineering and Knowledge Engineering Issues in Bioinformatics," *Handbook of Software Engineering and Knowledge Engineering, Vol. 1, Fundamentals*, Chapter 30, S. K. Chang, Ed., World Scientific Publishing Company, 2001, pp. 719-732.

Vincent Oria, Katherine G. Herbert, Viswanath Neelavalli, "An Automated Tool for Metadata Generation for Courseware-on-Demand," Technical Report submitted to NJ-ITOWER, December 2001.

Katherine G. Herbert, Huiyuan Shan and Jason T.L Wang, "Approximate Searching in Phylogenetic Databases," *The Proceedings of the Atlantic Symposium on Computational Biology and Genome Information Systems & Technology,* March 2001, pp. 140-143.

Katherine G. Herbert, "Data Integration in Biological Databases," Conference Presentation, *The Atlantic Symposium on Computational Biology and Genome Information Systems and Technology*, Duke University, Raleigh, North Carolina, USA, September, 2003.

Katherine G. Herbert, "TreeRank: A Similarity Measure for Nearest Neighbor Searching in Phylogenetic Databases," Conference Presentation, *The IEEE 15th International Conference on Scientific and Statistical Database Management*, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA, July 2003.

Katherine G. Herbert, Conference Presentation, "A Structure-Based Search Engine for Phylogenetic Databases," *The IEEE 14th International Conference on Scientific and Statistical Database Management*, University of Edinburgh, Edinburgh, Scotland, July 2002.

Katherine G. Herbert, Conference Presentation, "ATreeGrep: Approximate Searching in Unordered Trees," *The IEEE 14th International Conference on Scientific and Statistical Database Management*, University of Edinburgh, Edinburgh, Scotland, July 2002.

Katherine G. Herbert, Conference Presentation, "Approximate Searching in Phylogenetic Databases," *The Atlantic Symposium on Computational Biology and Genome Information Systems and Technology*, Duke University, Raleigh, North Carolina, USA, March 2001.

Since I have begun this journey, many people have affected my life and this work.

This dissertation is dedicated to them, those who have been my teachers, my mentors and my friends.

Also, I wish to dedicate this work to Mr. Connor Joseph Bracken and Miss Leila Grace Bracken. Your love of life, enthusiasm and hugs has been a source of inspiration and comfort for me. I love you both very much.

ACKNOWLEDGMENT

First, I would like to acknowledge and thank Dr. Jason T.L. Wang for his guidance over the past five years and his faith in me. Dr. Wang has given me opportunities I have only dreamed of and has been a true teacher to me. Words cannot express my gratitude towards him. Without him, this work would have not been possible.

I would also like to acknowledge and thank the member of my thesis proposal and dissertation committees Dr. Narian H. Gehani, Dr. James A.M. McHugh, Dr. Frank Shih, Dr. Vincent Oria and Dr. Michael Yin. Each of you who have served on these committees has been an inspiration to me. I am indebted to them for their service as well as their guidance.

Moreover, I would like to thank all of the external scientists and reviewers who have critiqued this work and given me feedback. Thank you especially to Dr. Dennis Shasha, Dr. William H. Piel, Dr. Roderick Page, Dr. Cathy H. Wu, Dr. Helen Berman and Dr. John Westbrook.

Special thanks also go to the faculty and administrative staff in the College of Computing Sciences and NJIT, especially Lorietta Barnes, Angel Bell, Michele Craddock, Artur Czumaj, Annette Damiano, Edith Frank, Robert Friedman, Clarisa Gonzalez-Lenahan, Andrew Hrechak, Jonathan Kapaleau, Kathy James, Ronald Kane, Marc Ma, Mariann McCoul, Thomas Moore, Mojgan Mohatamashami, David Nassimi, George Olsen, Wallace Rutkowski, Marc Sequiera, Alexander Thomasian, Michael Tress, Wenping Yang, the Department of Graduate Studies, the University Library Staff as well as many individuals throughout the New Jersey Institute of Technology Campus. Each of these staffs has aided me in numerous ways through my years at NJIT.

Moreover, my heartfelt appreciation goes to all of the students who have been involved with the CS Ph.D. program and the masters and Ph.D. students in the Data and Knowledge Engineering Laboratory especially Sen Zhang, Huiyuan Shan, Jiawei Liu, Xiang Hu, Vishu Neelivalli and Peng Fu.

Finally, I would like to acknowledge and thank the members of my family and friends who have been patient, kind and supportive throughout this entire process. Thank you to, my immediate family who has shown me patience, love and kindness throughout the process of my dissertation: Evelyn Herbert, John Herbert, William Herbert, Christine Bracken, Brian Bracken, Connor Bracken, Leila Bracken, and Grace Kahn. Also, thank you to my extended family for your enthusiasm about my work. Moreover, thank you to my very dear friends Alyssa Buxbaum, Alicia Drobes, Enzo Fonzo, Larry Green, Lynn Huang, Richard Lenert, Jason Ligo (for being a constant support through too many things to list), Brian McGuire, Godfrey Navarro (especially for proofreading), Diana and Matt O'Connell, Dr. Eileen Poiani, Colleen Roche, Dr. Robert Siegfried, Niraj and Arpita Shah, Michael Stone (especially for proofreading and tech support), Claudia Strano and the Strano Family, Kanwal Ullah and Baapen Yan.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

Chapter **Page**

TABLE OF CONTENTS (Continued)

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

When most databases are designed, the schema is perfected and issues concerning the data are usually accounted for. 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. For example, in an active database, if multiple users are adding data to the database, a number of misspellings or incorrect transcriptions can occur. Also, 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. Unless all of the databases use the exact same schema, the data from these multiple databases must be made to conform or interact with the new system.

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 analyzes 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 [108]. Two methods for managing data quality are data cleaning and data integration. Primarily, data cleaning addresses the issues of "detecting and removing errors and inconsistencies from data in order to improve the quality of the data" [82]. There have been a number of commercial tools developed to perform data cleaning, but most have only been able to provide limited services. Moreover, most operate specifically on relational databases containing text data. While the effectiveness of these tools is questionable, no tool has been specifically developed for use with biological databases. Data integration consists of the problem of "combining data residing at different sources and providing the user with a unified view of the data" [59].

This dissertation will discuss the need for biological data cleaning integration and propose a possible framework for performing data cleaning on biological databases. It will then show how this framework, BIO-AJAX, can be 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 analyzed for specific databases, there has been very little work regarding data cleaning when referring to biological and evolutionary databases. Moreover, it will also discuss data integration and its methods for helping to map biological data so that the user has a streamline view of the data.

Motivation

Biological data, specifically phylogenetic 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 standardize the data for further computation and improve the quality of the data for searching. Data integration is also an important function necessary for analyzing biological data. The very core purpose for most biological databases is to create repository, integrating work from numerous scientists.

Phylogenetic data encapsulates these problems. It is a complex data set that consists for processed wet lab results, data obtained through knowledge discovery, structural data and metadata. Moreover, phylogenetic studies also have multiple applications from Tree of Life studies to biomedical investigations. Using phylogenetic data to demonstrate these problems, data cleaning and integration can be used to solve the following problems:

- Standardizing the format of phylogenetic data when performing operations upon them
- Cleaning and modifying legacy data
- Standardizing nomenclature
- Finding duplicate trees and records within the dataset
- Removing duplicates, if necessary
- Clustering similar trees and records
- Merging similar records
- Finding anomalous trees and data

Each of these problems can pose serious problems for phylogenetic databases. First, many databases have standardization issues. These issues can be caused by a number of problems. 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, which is data that has been contained within the database for a long period of time, there can also be standardization problems. For example, one common problem with legacy data is that legacy data can be stored in a different schema or format than more recent data. If a database is a few years old, the format in which the data is kept within the database can change. This can be due to a number of reasons, especially for biological data. 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 older 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 from these various formats.

Another large problem facing phylogenetic databases is the nomenclature issue. Nomenclature refers to a naming system. There are various types of naming systems, but with phylogenetic data, the primary nomenclature issues concern the naming schemes related to how the species are classified. Currently, the most pervasive nomenclature methodology within biology for species is the Linnaean Hierarchy. The Linnaean Hierarchy is the method developed by Carl Linnaeus in 1788. Since this classification system has been adopted, a number of problems have been observed. Many phylogenetic researchers find this system of classification inadequate and have proposed other nomenclature methods. The Linnaean Hierarchy primarily depends upon naming species based on the observations of the species and classifying the species based on those 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 not well-known 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 separate Linnaean names while being the same species. Also, by using Linnaean names exclusively, this can limit the users of a phylogenetic database tool. Many potential users, possibly recreational users, may be unfamiliar with the various scientific names associated with an species. They may prefer to use their native language's particular vernacular description of the species. If the database limits the user only to the scientific names, or if there is not some mechanism for having all of these naming methods interact, such as through a thesaurus, this can limit any users ability to search the database effectively. For phylogenetic databases, this can limit the amount of information a user can obtain as well as limit the descriptiveness of the user's query. This creates the problem of standardizing 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 [40].

Another issue that can exist in phylogenetic databases is duplication of 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. In any case, 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 to be 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 (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 used to create 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 behavior 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`), a manually curated phylogenetic tree repository, can be generalized to biological databases. The next chapter introduces some of the biological concepts that make the data cleaning and integration problem interesting for biological data and gives a framework for cleaning biological data. 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 standardizing 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 [35]. 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 dissertation discusses the aforementioned problems and introduces new methods to help preserve biological data quality. The next chapter, Information Retrieval and Data Mining, discusses the state of the art in information retrieval and data mining, especially concerning the World Wide Web. Following that, the Biological Data Background chapter mentions phylogenetic data and current issue facing it. The Data Cleaning chapter reviews the state of the art in data cleaning, a key component of data quality. The next chapter introduces BIO-AJAX and formalizes the conceptual framework. Chapter 6 discusses the state of the art in data integration as well as establishes a new three-tiered integration interpretation specifically for biological data repositories. Chapter 7 and 8 demonstrate practical implementations of BIO-AJAX. Chapter 7 exhibits BIO-AJAX for TreeBASE, a phylogenetic nomenclature cleaning tool that employs mediator integration. Chapter 8 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, Chapter 9 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.

CHAPTER 2

INFORMATION RETRIEVAL AND DATA MINING

2.1 Introduction

With both commercial and scientific data sets growing at an extremely rapid rate, methods for retrieving knowledge from this data in an efficient and reliable manner are constantly needed. To do this, many knowledge discovery techniques are employed to analyze these large data sets. Generally, knowledge discovery is the process by which data is cleaned and organized, then transformed for use in pattern detection and evaluation tools and then visualized in the most meaningful manner for the user [39].

Two areas of research, information retrieval (IR) and data mining (DM), are used to try to manage these data sets as well as gain knowledge from them. Data mining concentrates on finding and exploiting patterns found within a given data set to gain knowledge about that data set.

As databases developed and became larger and more complex, the need to extract knowledge from these databases became a pressing concern. Data mining uses various algorithms that extract patterns from the data to gain knowledge about the data set. It borrows techniques from statistics, pattern recognition, machine learning, data management and visualization to accomplish the pattern discovery task. Information retrieval is the study of techniques for organizing and retrieving information from databases [85]. Modern information retrieval concerns itself with many different types of databases. It studies returning information matching a user's query that is relevant in a reasonable amount of time. It also focuses on other complex problems associated with a static query that will be needed time and time again.

This chapter explores both the topics of data mining and information retrieval. It discusses how these two approaches of obtaining knowledge from data can work in a complementary manner to create more effective knowledge discovery tools. The chapter introduces common application of knowledge discovery tools where these approaches work together, namely search engines. Finally, it addresses future work in data mining and information retrieval.

2.2 Information Retrieval

As mentioned previously, information retrieval investigates problems that are concerned with organizing and accessing information effectively. This is a broad area of research that currently encompasses many disciplines. Here, the chapter primarily focuses on text information retrieval, and then briefly mentions emerging areas such as web and multimedia information retrieval.

2.2.1 Text Retrieval Issues

Text retrieval systems usually need to perform efficient and effective searches on large text databases, often with data that is not well organized. Text retrieval is generally divided into two categories: problems that concentrate on returning relevant and reliable information to the user and problems that concentrate on organizing data for long-term retrieval needs. Concerning the first problem, methods here usually investigate techniques for searching databases based on a user query. The user can enter a query and the text retrieval system searches the database, returning results based on the user's query. These results can be ranked or ordered according to how close the text retrieval system feels the results satisfy the query. Another type of text retrieval system is one that is used for long-term information needs. These employ text categorization, text routing, and text filtering techniques to enhance the user's ability to query the database effectively. These techniques essentially preprocess a portion of the querying process, whether it is classifying the text or creating a user profile to better semantically query the database or use filters on the database before beginning the search [85].

Text retrieval systems have many issues that they must address in order to effectively perform searches on a database, specifically text databases. Many of these issues result from the vernacular usage of words and phrases within a given language as well as the nature of the language. Two prominent issues that need to be addressed by text retrieval systems related to this problem are synonymy and polysemy.

- *Synonymy* refers to the problem of when words or phrases mean similar things. This problem sometimes is solved and usually results in a text retrieval system needing to expand upon a query, incorporating a thesaurus to know which words or terms are similar to the words or terms in the user's query. This allows the system to return results that might be of interest to the user but would normally be returned for another word that has a similar meaning to the word or phrase used within the query.

- *Polysemy* refers to when one word or phrase has multiple meanings [85]. Work to address this problem has included creating user profiles so that the text retrieval system can learn what type of information the user is generally interested in as well as semantic analysis of phrases within queries [52].

Other common problems that text retrieval systems must be concerned with are phrases, object recognition and semantics. Phrases within languages tend to have a separate meaning from what each individual word in the phrase means. Many text retrieval systems use phrase based indexing techniques to manage phrases properly [22]. Object recognition usually concerns itself with a word or phrase. These word phrases usually have a specific meaning separate to itself from the meaning of the individual words. For example, the word "labor" means to work and the word "day" refers to a period of time. However, when these two words are placed next to each other to form "Labor Day", this refers to a holiday in September in the United States. Common parts of sentences that are considered objects are proper nouns, especially proper names, noun phrases and dates.

A text retrieval system that can manage objects sometimes uses pattern recognition tools to identify objects [85]. All of these problems can generally be thought of by considering how the word or phrase is used semantically by the user.

2.2.2 Text Retrieval Methods

To address these problems, there are some common practices for processing and filtering data that help text retrieval tools to be more effective. One very common practice is to use inverted files as an indexing structure for the database the tools search upon. An inverted file is a data structure that can index key words within a text. These keywords are organized so that quick search techniques can be used. Once a keyword is found within the indexing structure, information is retained about the documents that contain this keyword and those documents are returned to fulfill the query.

Figure 2.1 illustrates the general concept behind inverted files. In this figure, there is an organized structuring of the keywords. This structuring can be formed through using various data structures, such as the B-tree or a hash table. These keywords have references or pointers to the documents where they occur frequently enough to be considered a content-bearing word for that document. Deciding whether or not a word is content bearing is a design issue for the information retrieval system. Usually, a word is considered to be content-bearing if it occurs frequently within the document. Algorithms such as Zipf's law [117] or taking the term frequency with respect to the inverse document frequency (tf.idf) [86] can be used to determining whether or not a word is a content-bearing word for a document.

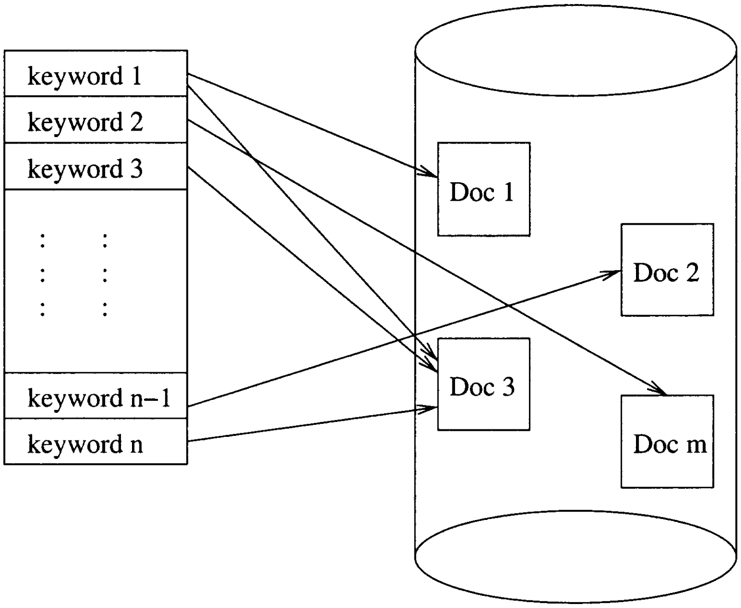


Figure 2.1 A general diagram for an inverted file.

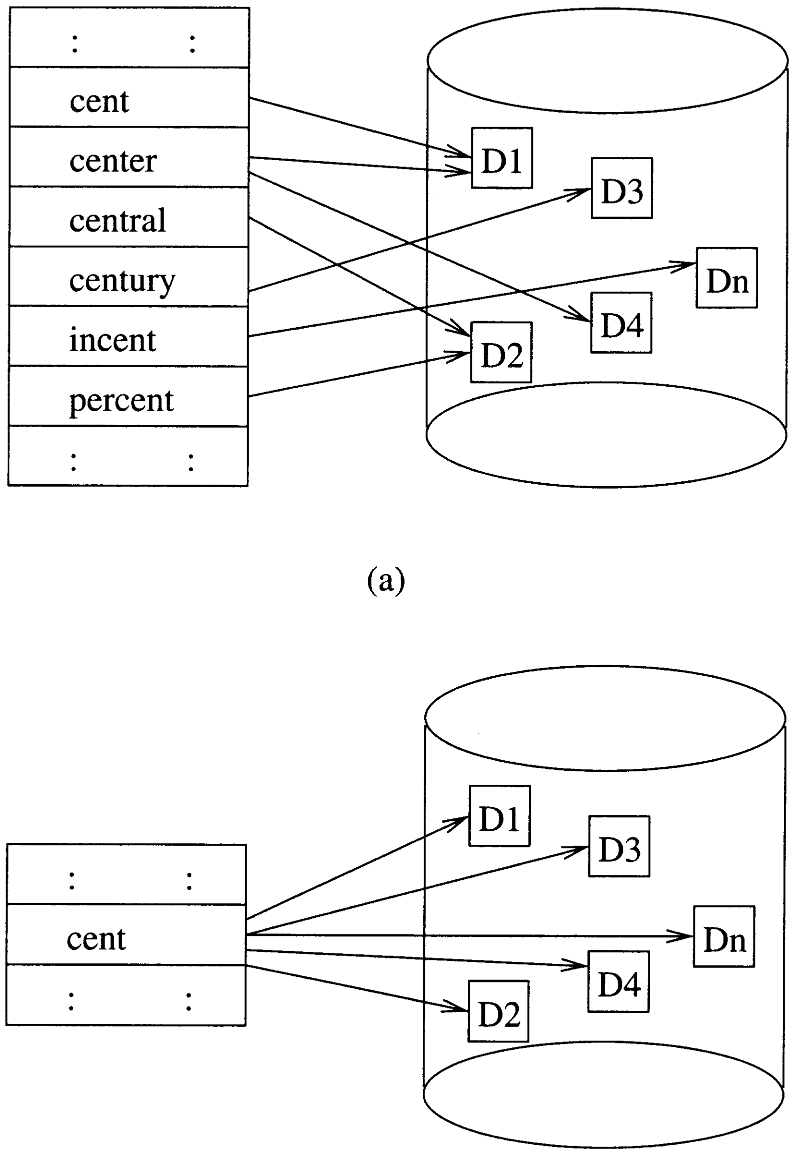
Other effective methods include stopword lists, stemming and phrase indexing. Stopword lists, commonly seen when searching the World Wide Web using Google, exploit the idea that there are great occurrences of common words, such as "the", "a", and "with", within documents in a given text database. Since these words rarely add to the meaning of the query, they are disregarded and filtered out of the query so that the text retrieval tool can concentrate on the more important words within the query. Stemming utilizes the concept that many words within a query can be a variation on tense or case of another word. For example, the word "jumping" has the root word "jump" in it. The concepts related to "jumping" and "jump" are very similar. Therefore, if a query requests information about "jumping", it is highly likely that any information indexed for "jump" would also interest the user. Stemming takes advantage of this idea to make searching more efficient. Stemming can help improve space efficiency as well as help generalize queries. Generalized queries help to ensure documents that the user may want but might not have been included within the search results because of the wording of the query will be included. However, this can also lead to false positives if the stemming algorithm does not process a word properly [30, 84].

For example, Figure 2.2 demonstrates some of the issues stemming faces. Figure 2.2(a) represents how an index containing the words “cent”, “center”, “central”, “century”, “incent”, and “percent” might be organized. Figure 2.2(b) demonstrates how each of these words can be reduced to the root “cent”. Notice, in the first figure, each word indexes different documents. However, in the second figure, all of the documents that were reduced to the root “cent” are now indexed by cent. Stemming, while it might return related documents to the query, can also return many unrelated documents to the user. Also, while all of the terms in this figure can be reduced to the root “cent”, it is not appropriate in some cases to do so. In the case of “percentage” and “incent”, the issue of whether or not a prefix should be stripped arises. In this example, the prefixes are stripped off to demonstrate the problems of stemming. However, generally, prefixes are not stripped to reduce a word to its root since many prefixes change the meaning of the root.

2.2.3 Text Retrieval Systems and Models

Applying the above concepts to help filter and retrieve meaningful data, there are many methods for creating text retrieval systems. The most popular one is the Boolean keyword text retrieval system. In this system, the user enters a series of keywords joined by Boolean operators such as “and” or “or”. These systems can be extended to employ ranking of search results as well as the ability to handle wildcard or “don’t care” characters [91, 118, 119].

One popular model for text retrieval systems is the vector-space model [86, 87]. In vector-space models, documents are viewed as vectors in N-dimensional space, where N is the number of keywords within the database. The values within the vector represent whether a particular keyword is present in the given document. These values



(a)

(b)

Figure 2.2 The effect of stemming on the inverted file index. (a) represents an inverted file that does not use stemming. (b) represents an inverted file that uses stemming.

can be as simple as 0 if the keyword is not present within the document or 1 if it is present. Also, the text retrieval system can use functions where the resulting value represents the importance of that keyword within the document. When querying the system, the user's query is transformed into a vector and then that vector is compared with the vectors within the database. Documents with similar vectors are returned to the user, usually ranked with respect to how similar the document's vector is to the original query. While this technique does allow for effective similarity searches, the dimensionality of the vectors can increase greatly depending upon the number of keywords.

Other popular text systems include probabilistic models as well as the employment of machine learning and artificial intelligence techniques. Probabilistic models test how well a document satisfies a user query. These techniques can employ Bayesian networks to represent both the document and the query. Machine learning and artificial intelligence techniques use natural language processing, rule-based systems and case-based reasoning for information retrieval in text-based documents.

2.2.4 Web and Multimedia Information Retrieval

While there is a great amount of interest in the text-base methods previously mentioned, there is also currently a lot of research within the areas of Web-based information retrieval and multimedia information retrieval. Web-based information retrieval can best be seen in the techniques used to index websites for search engines as well as the techniques used to track usage [18, 52]. Issues concerning this type of information retrieval will be reviewed later in this chapter. Multimedia information retrieval addresses issues involved with multimedia documents such as images, music and motion pictures. Unlike text, multimedia documents use various formats to display information. Since this information is in different formats, before any retrieval algorithms are used on it, the documents must be mapped to a common format. Otherwise, there will be no standard representation of the document, making retrieval harder. Moreover, since most information retrieval algorithms apply to text documents, these algorithms either need to be modified or new algorithms have to be developed to perform retrieval. Also, most tools can only process information about the format of the document, not its content [68].

Besides issues concerning the data that are comprised in a multimedia database, other issues concerning the query also cause great difficulties. Since the query would be for multimedia information, naturally the user might want to use a multimedia format to author the query. This creates problems with user authoring tools. For example, if someone is searching for a particular piece of music and only knows a couple of notes from the song (and knows nothing about the title or artist), he or she has to enter those notes. This creates issues concerning how does the user enter these notes as well as how are these notes matched against a database of songs. Most multimedia databases allow only keyword search thereby eliminating the authoring problem. Databases that allow only keyword search usually use whatever metadata about the images within the database (such as captions and file names) to index the data. The query is then matched against this data. A good example of this type of search tool is Google's Image Search (`http://www.google.com`). However, there are many research projects underway to allow users to author multimedia documents (such as images or segments of music) as queries for databases [58].

Currently there are many efforts for developing techniques to both process and retrieve multimedia data. These efforts combine numerous fields outside of information retrieval including image processing and pattern recognition. Current popular techniques use relevance feedback as well as similarity measures such as Euclidean distance for multimedia information retrieval. Relevance feedback is a technique used where the user continually interacts with the retrieval system, refining the query until he or she is satisfied with the results of the search. Similarity measures are used to match queries against database documents. However, current similarity measures, such as Euclidean distance and cosine distance for vector models, are based on the properties of text retrieval. Therefore, relevance feedback practices have performed better than the similarity measures [58]. One project by the Motion Picture Expert Group (MPEG) is called MPEG-7. MPEG-7 tries to create models for various types of multimedia documents so that all of the information contained within the documents can be specified through metadata. This data can then be searched through the usual text retrieval methods [72, 73].

2.2.5 Evaluating IR Systems

Information retrieval systems are evaluated based on many different metrics. The two most common metrics are recall and precision.

- *Recall* measures the percentage of relevant documents retrieved by the system with respect to all relevant documents within the database. If the recall percentage is low, then the system is retrieving very few relevant documents.

- *Precision* describes how many false hits the system generates. Precision equals the number of relevant documents retrieved divided by the total number of documents retrieved. If the precision percentage is low, then most of the documents retrieved were false hits.

Most IR systems experience a dilemma concerning precision and recall. To improve a system's precision, the system needs strong measures for deciding whether a document is relevant to a query. This will help minimize the false hits, but it will also affect the number of relevant documents that are retrieved. These strong measures can prevent some important relevant documents from being included within the set of documents that satisfy the query thereby lowering the recall.

Besides precision and recall, there are other measures that can be used to evaluate the effectiveness of an information retrieval system. One very important evaluation measure is ranking. *Ranking* refers to the evaluation techniques by which the search results are ordered and then returned to the user, presented in that order. In ranking, a rating is given to the documents that the information retrieval system considers a match for the query. This rating reflects how similar the matched document is to the user's query. One of the most popular algorithms for ranking documents on the World Wide Web is PageRank [6, 77]. PageRank, developed by Page *et al.*, is for understanding the importance for documents retrieved from the Web. It is similar to the citation method of determining the importance of a document. Basically, in this algorithm, the relevance of a website to a particular topic is determined by how many well-recognized Web pages (Web pages that are known to be a reliable reference to other pages) link to that page. Besides precision, recall and ranking, which are based on system performance, there are also many measures such as coverage ratio and novelty ratio that indicate the effectiveness of the information retrieval system with respect to the user's expectations [53]. Table 2.1 summarizes some of these measures that can be used to determine the effectiveness of an information retrieval system.

2.3 Data Mining

Data mining refers to the extraction or discovery of knowledge from large amounts of data [39, 105]. Other terms with similar meaning include knowledge mining, knowledge extraction, data analysis and pattern analysis. The main difference between information retrieval and data mining is their goals. Information retrieval helps users search for documents or data that satisfy their information needs [7]. Data mining goes beyond searching; it discovers useful knowledge by analyzing data correlations using sophisticated algorithmic techniques. Knowledge may refer to some particular patterns shared by a subset of the dataset, some specific relationship among

Table 2.1 Measures for Evaluating Information Retrieval Systems

	Measure	*Purpose*
(i)	Precision	Describes the number of false hits.
(ii)	Recall	Measures percentage of relevant documents retrieved with respect to all relevant documents within the database.
(iii)	Coverage	Measures the number of relevant documents retrieved the user was previously aware of.
(iv)	Citation	Measures importance of a document through the number of other documents referencing it.
(v)	Novelty	Measures the number of relevant documents retrieved the user was not previously aware of.
(vi)	PageRank	This is similar to the citation measure, except for Web documents.

a group of data items, or other interesting information that is implicit or not directly inferable.

Data mining is an interdisciplinary field contributed to by a set of disciplines including database systems, statistics, machine learning, pattern recognition, visualization, and information theory. As a result, taxonomies of data mining techniques are not unique. This is due to the various criteria and viewpoints of each discipline involved with the development of the techniques. One generally accepted taxonomy is based on the data mining functionalities such as association rule mining [2], classification [14], clustering [5], and concept description. To be a comprehensive and effective data mining system, the above functionalities must be implemented within the system. These functionalities also give a portal to the understanding of general data mining system construction.

2.3.1 Concept Description

The explosive increase of data volume, especially large amounts of data stored in great detail, requires a succinct representation for the data. Most users prefer an overall picture of a class of data so as to distinguish it from other comparative classes. On the other hand, the huge volume of data makes it impossible for a person to give, intuitively, such a concise while accurate summarization for a given class of data. However, there exist some computerized techniques to summarize a given class of data in concise, descriptive terms, called concept description [18, 19]. These techniques are essential and form an important component of data mining.

Concept description is not simply enumeration of information extracted from the database. Instead, some derivative techniques are used to generate descriptions for characterization and discrimination of the data. According to the techniques used to derive the summary, concept description can be divided into characterization analysis and discrimination analysis. *Characterization analysis* derives the summary information from a set of data. To do characterization, the data generalization and summarization-based method aims to summarize a large set of data and represent it at a higher, conceptual level. Usually, attribute-oriented induction is adopted to guide the summarization process from lower conceptual level to higher one by checking the number of distinct values of each attribute in the relevant set of data. For example, Table 2.2 shows the original data tuples in a transactional database for a chain company. If some generalization operation regarding the geographical locations of stores has already been established, then the store ID in the location field can be replaced by a higher level description, namely geographical areas. In addition, generalization can be done on the time field by replacing it with a higher level concept, say month. Table 2.3 shows generalized sales for the database in Table 2.2 where the generalizations are performed with respect to the attributes "time" and "location".

Table 2.2 Original Data in a Transactional Database

	Item	*Unit Price*	*Time*	*Payment*	*Loc.*	*Quant.*
(i)	printer	$45.00	14:02 7/5/2002	Visa	0089	1
(ii)	scanner	$34.56	11:09 8/1/2002	Cash	0084	1
(iii)	camcorder	$489.95	13:00 7/14/2002	Master	0100	1
(iv)	⋮	⋮	⋮	⋮	⋮	⋮

Discrimination analysis puts emphasis on the distinguishing features among sets of data. Discrimination analysis can be accomplished by extending the techniques

Table 2.3 Generalized Sales for the Same Transactional Database in Table 2.2

	Item	*Unit Price*	*Time*	*Payment*	*Loc.*	*Quant.*
(i)	printer	$45.00	July, 2002	Visa	Essex	1
(ii)	scanner	$34.56	August, 2002	Cash	Hudson	1
(iii)	camcorder	$489.95	July, 2002	Master	Essex	1
(iv)	⋮	⋮	⋮	⋮	⋮	⋮

proposed for characterization analysis. For instance, by performing the generalization process among all data classes simultaneously and synchronously, the same level of generalization for all of the classes can be reached, making the comparison feasible. In previous examples, it was assumed the attributes selected for characterization or discrimination are always relevant. However, in many cases, not all of the attributes are relevant for data characterization or comparison. Analytical characterization techniques are one kind of attribute relevance analysis. They are incorporated into data description or comparison to identify and exclude those irrelevant or weakly relevant attributes.

Concept description tries to capture the overall picture of a class of data by inducing the important features of it through conceptual generalization or comparison with a class of comparative data. By grasping the common features presented by the data class as a whole, it looks the class of data as an entirety while ignoring the relationship among its component items. However, in many cases, exploring the relationship within component items is valuable. This forms another important data mining process: association rule mining.

2.3.2 Association Rule Mining

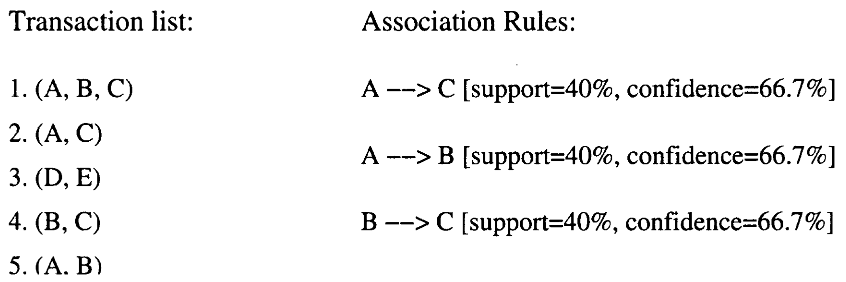
Association rule mining [3, 4, 67] is the process of finding interesting correlations among a large set of data items. For example, the discovery of interesting association relationships in large volumes of business transactions can facilitate decision making in marketing strategies. The general way of interpreting an association rule is that the appearance of the item(s) on the left hand side of the rule implies the appearance of those item(s) on the right hand side of the rule.

There are two parameters to measure the interestingness for a given association rule: support and confidence. For instance, consider the following association rule discovered from a transaction database:

$$B \rightarrow C \ [support = 30\%, confidence = 66\%]$$

The usefulness of an association rule is measured by its support value. Given the above rule, it means within the whole transactions of the database, 30% transactions contain both items B and C. The confidence value measures the certainty of the rule. Again for the above rule, it means for all those transactions containing B, 66% of them also contain C. Figure 2.3 shows an example of finding association rules from a set of transactions. For rule $A \rightarrow C$, the number of transactions containing both A and C is 2, so the support for this rule is 2 divided by the total number of transactions (5), which is equivalent to 40%. To calculate confidence, first find the number of transactions containing A is 3, resulting in a confidence of 66.7%.

An acceptable or interesting rule shall have its two parameter values greater than a user-specified threshold. These two parameters are intuitively reasonable for measuring the interestingness of an association rule. The support parameter



Transaction list:

1. (A, B, C)
2. (A, C)
3. (D, E)
4. (B, C)
5. (A. B)

Association Rules:

A --> C [support=40%, confidence=66.7%]

A --> B [support=40%, confidence=66.7%]

B --> C [support=40%, confidence=66.7%]

Figure 2.3 A simple example of finding association rules.

guarantees that there are statistically enough transactions containing the items appearing in the rule. The confidence parameter implies the validness of the right hand side given the left hand side of the rule, with certainty.

Given the two parameters, support and confidence, finding association rules requires two steps. First, find all frequent itemsets, which contain all the itemsets so that, for each of them, its number of appearances as a whole in the transactions must be greater than the support value. Next, generate association rules that satisfy the minimum support and minimum confidence, from the above frequent itemsets.

The well-known Apriori [3, 4, 67] data-mining algorithm can demonstrate the principles underlying association rule mining. Apriori is a classic algorithm to generate all frequent itemsets for discovering association rules given a set of transactions. It iteratively scans the transaction set to find frequent itemsets at one particular size at a time. During each iteration process, new frequent candidate itemsets with size one larger than the itemsets produced at previous iteration are generated; and the acceptable itemsets are produced and stored through scanning the set and calculating the support value for each of the candidate itemsets. If no new frequent itemsets can be produced, Apriori stops by returning all itemsets produced from every iteration stage.

Given the frequent itemsets, finding association rules is straightforward. For each itemset, divide the items in it into two subsets with one acting as the left hand side of the association rule and the other as the right hand side. Different divisions will produce different rules. In this way, all of the candidate association rules can be found. It is obvious that each association rule satisfies the requirement of minimum support. Association rules can be further generated by verifying their confidence values.

2.3.3 Classification and Prediction

In many cases, making a decision is related to constructing a model, such as a decision tree [71], against which unknown or unlabeled data could be categorized or classified into some known data class. For example, through the analysis of the customers purchase behavior associated with age, income level, living area and other factors, a model can be established to categorize customers into several classes. With this model, new customers can be classified properly so that an appropriate advertising strategy and effective promotion method could be set up for maximizing profit.

Classification is usually associated with finding a known data class for the given unknown data, which is analogous to labeling the unlabeled data. Therefore, the data values under consideration are always discrete and nominal. On the other hand, prediction aims to manage continuous data values by constructing a statistical regression model. Intuitively, a regression model tries to find a polynomial equation in the multidimensional space based on the given data. The trends presented by the equation give some possible predictions. Typical applications include investment risk analysis and economic growth prediction.

In the past, several classification approaches have been developed. The major models include decision tree induction, Bayesian classification, Bayesian belief networks and neural network classification [110]. Even though each model has its particular trait, all of them share a common two-step processing feature: a training stage and a classification stage. During the training stage, a model describing a predetermined set of data classes is established through analyzing database tuples comprised by attribute values. These tuples constitute the training data set. The acceptability of the model is measured in the classification stage where another data set, called testing data set, is used to estimate the accuracy of the classification. If the model passes the classification stage, it means its classification accuracy is acceptable and is ready to be used for classifying future data tuples or objects whose class labels are unknown.

In regard to prediction, the available regression techniques include linear regression, nonlinear regression, logistic regression and Poisson regression [49]. Linear regression attempts to find a linear equation to represent the trend shown in the given database. Nonlinear regression uses a polynomial equation to represent the trend, instead of a linear equation, showing higher accuracy in those cases of complex trend prediction. Logistic regression and Poisson regression are also called generalized regression models, which can be used to model both contiguous and discrete data.

As described above, classification starts with a set of known labeled data and its training stage is guided by the labeled data. This kind of training or learning is called "supervised learning", where both the label of each training datum and the number of data classes to be learned are known. On the other hand, there exist many cases in which the knowledge about the given set of data is very limited. Neither is the label for each datum known nor has the number of data classes been given. Clustering, known as "unsupervised learning", is aimed to handle those cases.

2.3.4 Clustering

Clustering is the process of grouping data objects into clusters without prior knowledge of the data objects [51, 52, 111]. It divides a given set of data into groups so that objects residing in the same group are "close" to each other while being far away from objects in other groups. Figure 2.4 illustrates the general concept underlying

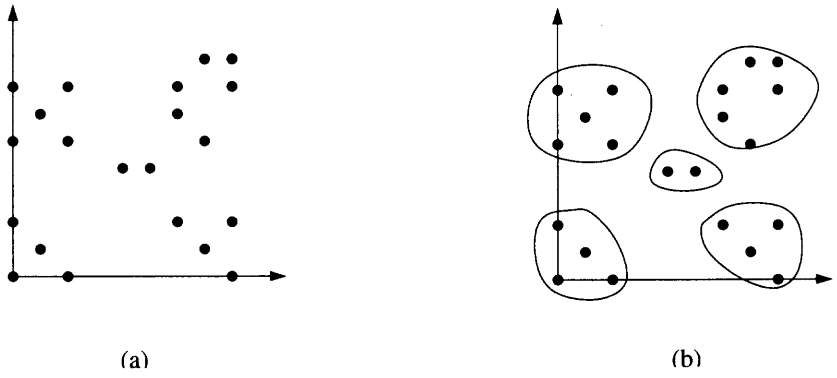


Figure 2.4 Clustering example. (a) A set of spatial points. (b) A possible clustering for the spatial points.

clustering. It has shown object-dense regions, represented as point sets, are found and objects are clustered into groups according to the regions.

The objective of clustering is to enable one to discover distribution patterns and correlations among data objects by identifying dense versus sparse regions in the data distribution. Unlike classification, which requires a training stage to feed predetermined knowledge into the system, clustering tries to deduce knowledge based on information gained from the data through the clustering analysis. Clustering analysis has a wide range of applications, including image processing, business transaction analysis, and pattern recognition.

The "learning from nothing" feature poses a set of typical requirements for an effective and efficient clustering analysis. These requirements, as discussed in [39], include scalability, capability of dealing with different types of data, ability to cope with noisy and high dimensional data, being able to be guided by clustering constraints and the capability to cluster arbitrary shapes. To meet these requirements, researchers have proposed many clustering algorithms by taking advantage of the data under analysis and the characteristics of the application. The major categorization of clustering methods could be: partition methods [64], hierarchical methods [51] and grid-based methods [109].

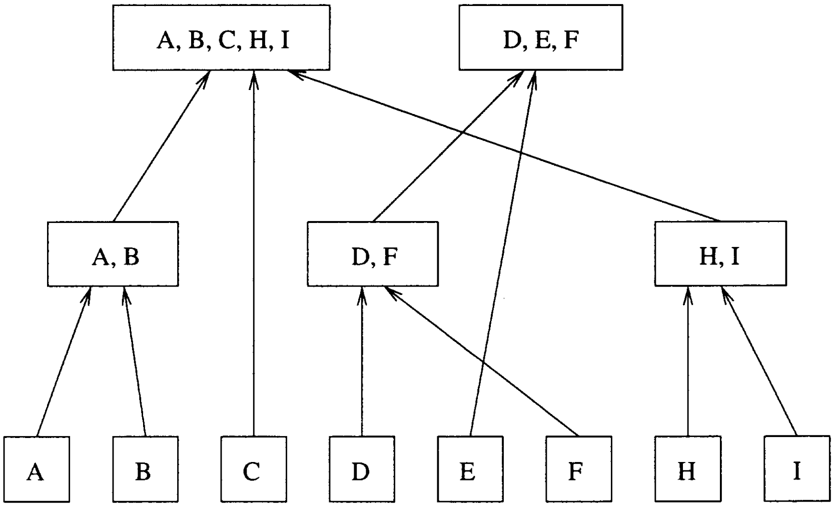


Figure 2.5 Agglomerative clustering.

The well-known k-means algorithm [64] and its variation k-medoid [51] are two partition methods which accept n data objects and an integer k, and then divide the n objects into k groups satisfying the following two conditions. First, each group must contain at least one object. Second, each object must belong to exactly one group. During the clustering, partition methods adopt iterative relocation techniques to try to find a different, more "reasonable" group for each data object and move data objects between groups until no group change occurs.

Hierarchical methods, such as agglomerative clustering, adopts a bottom-up strategy for the tree constructing. As shown in Figure 2.5, the leaf nodes are original objects. The clustering process goes from the bottom up along the tree with each internal node representing one cluster. On the other hand, divisive clustering [51] uses top-down tactics to accomplish the same goal.

Both density-based methods and grid-based methods can handle arbitrary shape clustering. Density-based methods [28] accomplish this by distinguishing object-dense from object-sparse regions. On the other hand, grid-based methods use a multidimensional grid data structure to accommodate data objects. Through the manipulation on the quantized grid cells, data objects are clustered. Model-based methods assume that the data are generated by a mixture of underlying probability distributions, thus the goal of clustering becomes finding some mathematical model to fit the given data.

2.4 Integrating IR and DM Techniques into Modern Search Engines

With the development of the World Wide Web as well as developments in information retrieval and data mining, there are many applications in which one can exploit IR and DM techniques to help people discover knowledge they need. The most common instances of these applications tend to be in search tools. This section reviews popular uses of information retrieval and data mining concerning the World Wide Web.

2.4.1 Web Mining and Retrieval

One popular approach for trying to improve the recall of Web-based search engines is to employ data mining techniques to learn more about both the data retrieved as well as user preferences. Data mining techniques not only help in ensuring the results of the query are precise but also to help the user sort through the search results that match a query. By using both data mining and information retrieval techniques to analyze the data, two extremely effective methods of analysis can work together to provide users with the most relevant information they are searching for.

Currently, there are a lot of applications being developed where data mining is applied to Web information retrieval problems. These problems can be classified into certain groups: Web content mining, Web structure mining, and Web usage mining. Web content mining discovers useful knowledge within data on the World Wide Web. This analysis studies the content of websites as well as procedures for extracting and analyzing that content. Web structure mining looks at how various websites are related to one another. This analysis usually tries to discover the underlying connections of websites on the Internet (usually through the analysis of hyperlinks) so as to discover relationships and information about the websites. Finally, Web usage mining studies the behavior of a group of users with respect to the websites they view. From these studies, it can be observed what websites various groups of people with similar interests consider important. This section concentrates solely on Web content mining since this type of mining is what most users have direct experience with [69].

One of the most popular applications of Web content data mining is clustering. Clustering algorithms are ideal for analyzing data on the Web. The premise behind clustering is that given a data set, find all groupings of data based on some data dimension. As discussed before, clustering, unlike some other popular data mining techniques such as classification, does not require any mechanism for the tool to learn about the data. A couple of search engines that employ clustering to help users have meaningful and effective search experiences are surveyed below.

2.4.2 Vivisimo

Vivisimo (`http://www.vivisimo.com`) [102] is a meta-search engine that uses clustering and data mining techniques to help users have a more effective search experience. The search engine developed by the Vivisimo company offers users both on the Web and through enterprise solutions the ability to cluster information extracted by a search tool immediately or "on the fly" [102]. Concentrating on Vivisimo's Web-based search engine, this tool creates an extremely useful searching environment. In this Web search tool, the user can enter a query similarly to any popular search engine. When this query is entered, the Vivisimo search tool sends the query to its partner Web search tools. Some of Vivisimo's partners include Yahoo! (`http://www.yahoo.com`), GigaBlast (`http://www.gigablast.com`), DogPile (`http://www.dogpile.com`), MSN (`http://www.msn.com`), and Netscape (`http://www.netscape.com`). Once the results of the searches for the query on these search tools are complete, the results are returned to the Vivisimo search tool. Vivisimo then employs proprietary clustering techniques to the resulting data set to cluster the results of the search. The user can then search through the results either by browsing the entire list, as with most popular search tools, or by browsing the clusters created.

Besides the traditional Web-based meta-search, Vivisimo allows users to search specifying certain websites, especially news websites. For those users who want to search for very current information, this tool can search a specific website for that information. It also organizes the information categorically for the user. For example, a user can use the Vivisimo cluster tool to search a news website for current information he or she is interested in. However, the user can only specify websites for this type of search from a list Vivisimo provides for the user.

2.4.3 KartOO

Another search tool that searches the Web, using clustering techniques, is KartOO (`http://www.kartoo.com`) [50]. KartOO is also a meta-search engine, similar to Vivisimo's search tool. However, KartOO's visualization methods add a new dimension for users to a data set. Similar to Vivisimo, KartOO uses a number of popular search engines for the initial search. The tool allows the user to select which search engines are included within the search. Once the results are returned, KartOO evaluates the result, organizing them according to relevance to the query. The links most relevant to the query are then returned as results. When the results are presented, the results are represented in an interactive graph for the user. Each node, or "ball" of the graph, represents a website that was returned to KartOO as fulfilling the query. Each node is connected to other nodes through edges that represent semantic links between the websites modeled within the node. The user can then browse the graph looking for the information he or she is interested in. While browsing, if the user rolls the mouse over a node, information about the link it connects is displayed. When the user rolls the mouse over one of the semantic links, the user can elect to refine his or her search to purposely include that semantic information within the query by clicking on the plus "+" sign or purposely exclude that semantic information within the query by clicking on the minus "-" sign. If the user neither wants to include or exclude that information, he or she can take no action. Through this interaction with the semantic links, the user can refine his or her query in a very intuitive way. Moreover, with the graphical representation of the results, a user can see how various results are related to one another, thus identifying different clusters of results.

2.4.4 SYSTERS Protein Family Database

Looking at more domain specific search tools, many research websites are also employing data mining techniques to make searching on their websites more effect. SYSTERS Protein Family Database (`http://systers.molgen.mpg.de/`) [54] is another interesting search tool that uses both clustering and classification to improve upon searching their data set. The SYSTERS Protein Family Database looks at clustering information about biological taxonomies based on genetic information and then classifies these clusters of proteins into a hierarchical structure. This database can then be searched using a variety of methods [54].

At the core of this tool are the clustering and classification algorithms. To place a protein sequence into the database, first the database uses a gapped BLAST search, which is a sequence alignment tool, to find what sequences the protein is similar to. However, since the alignment is asymmetric, this step is only used to narrow down the possible sequences the original might be similar to. Next, a pair wise local alignment is performed, which the clustering of the protein sequences will be based upon. Since these are biological sequences, all of the sequences will have some measure of similarity. Sequences that are extremely similar are clustered together, creating superfamilies. These superfamilies are then organized hierarchically to classify their relationships. Users can then search this database on a number of key terms including taxon (organism) names and cluster identification terms such as cluster number and cluster size. From this search, information about the specific query term is returned as well as links to related information in the cluster. The user can then browse this information through traversing the links.

2.4.5 E-Commerce Systems

Besides search tools, Web content data mining techniques can be used for a host of applications on the World Wide Web. Many E-commerce sites use association rule mining to recommend to users other items they might possibly like based on their previous selections. Association rule mining allows sites to also track the various types of usage on their site. For example, on an E-Commerce site, information about the user's interactions with the site can help the E-Commerce site customize the experience for the user, improve customer service for the user and discover customer shopping trends [39]. Also, concerning financial issues, classification and clustering can be used to target new customers for products based on previous purchases. Data mining can also give companies insights into how well a marketing strategy is enticing customers into buying certain products [39].

2.5 Conclusion and Further Resources

Information retrieval and data mining are two very rich fields of computer science. Both have many practical applications while also having a rich problem set that allows researchers to continually improve upon current theories and techniques. This chapter has looked at some of these theories and applications. Information retrieval has evolved from a field that was initially created for the need to index and access documents into a robust research area that studies techniques for not only retrieving data but also discovering knowledge in that data. Data mining, while a much younger field, has evolved to explore intriguing relationships in very complex data. The future promises to be very exciting with the developments in multimedia information retrieval and data mining as well as the movement towards trying to understand semantic meanings with the data.

While this chapter introduces these topics, there are many other resources available to readers who wish to study specific problems in-depth. In the field of information retrieval, there are a number of introductory texts that discuss information retrieval very comprehensively. Some excellent introductory texts include Salton's *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer* [86], Korfhage's *Information Storage and Retrieval* [53] and Baeza-Yates' and Ribeiro-Neto's *Modern Information Retrieval* [7]. Also, there are several conferences in which state-of-the-art research results are published. These include the Text Retrieval Conference (TREC, `http://trec.nist.gov/`) and ACM's Special Interest Group on Information Retrieval (SIGIR) Conference. Concerning data mining, there are also a number of introductory texts to this subject; see, for example, Han's and Kamber's *Data Mining: Concepts and Techniques* [39]. In addition, there are a great amount of data mining resources available as well as technical committees and conferences. A major conference is ACM's Special Interest Group on Knowledge Discovery in Data (SIGKDD) Conference, among others.

CHAPTER 3

BIOLOGICAL DATA BACKGROUND

Biological databases, like most databases are subject to many problems. Like any other database, biological databases have data that does not conform to the original intent of the database, or "dirty" data. However, due to the nature of the biological data, usually these problems become complex thereby excluding the more traditional methods for solving the "dirty" data problems. Conceptually, a framework for cleaning and integrating biological data would need to address both the schema and the data issues that cause problems concerning the data quality within the database [82].

Concerning schema related problems, usually these problems can fall into a few categories. First, the database is part of a federated system or interacts with other databases to obtain data. This data may or may not use the same schema currently used by the database. Therefore, this data needs to be mapped onto the database's schema. Second, sometimes, an error is made in the design of the database's schema. Concerning biological databases, this can very easily happen since most schemas need to model complex biological and chemical structures as well as the metadata involved with the data. A method may be needed by a biological database to modify the schema of the database while preserving the data.

Mapping the new schema to the old schema can help to solve this problem. Also, this mapping can be viewed as a data integration problem. Since all data submitted to the repository needs to be included with the current dataset, the submission problem can be seen as an integration problem. Moreover, if the repository transitions to a new schema, the legacy data needs to be integrated to interact properly with the data in the new schema. In addition, if the repository interacts with other repositories, this becomes the classical data integration problem.

Since many of the biological fields are dynamic, with new knowledge being generated very quickly, schemas can either change or be inadequate for new data. If the database changes schemas to create a more powerful database, issues concerning how to integrate both the legacy schema and legacy data arise. Any data that conforms to the legacy schema will be missing some data that is expected by the new schema. It will not conform to the new. Therefore, the legacy data will need to be mapped to the new schema [82].

Besides the schema related problems, there are also numerous data problems. First, data can be entered into the database erroneously [82]. This data will need to be corrected. Due to the dynamic nature of these databases, and since these databases can house anything from protein structures to strings about the genetic composition of a species; it is highly possible duplicate record data occurs. This data will need to be first detected and then organized properly. Also, concerning the legacy data, it will need to be integrated properly into the database. This will require mapping it properly as well as using various detection mechanisms to discover if this data previously exists. Thirdly, much of the data within biological databases is generated through the use of computational tools. Errors can occur through in improper creation of a tool using an algorithm to create data. For example, phylogenetic data uses reconstruction algorithms applied to taxa data to generate phylogenetic tree. If the reconstruction algorithm is instantiated improperly, the data will be skewed towards that error. This can have serious consequences since, if the data is used for further purposes, the error will be propagated throughout any application using that data.

Over the past few years, the proliferation of biological databases, data banks and data warehouses has been remarkable. First, there are large databanks that act as repositories for genetic data and their mirror sites. Examples of these databases are GenBank [8, 9], the Nucleic Acids Sequences Database at the National Center for Biotechnology Information in the United States (http://www.ncbi.nlm.nih.gov/), and its partner databases at the EMBL Data Library [98] (http://www.ebi.ac.uk/ embl/index.html), and the DNA Data Bank of Japan [99] (http://www.ddbj.nig.ac.jp/). There are also specialized or "boutique" databases that specialize in specific functionalities. Examples of these databases include TreeBASE [81] (http://www.treebase.org/treebase/index.html), and Ribosomal Database Project II (RDP II) [21], (http://rdp.cme.msu.edu/html/).

These databases can also be curated in a number of different ways. Most are public repositories, where data can be submitted through the World Wide Web. Examples of this type of database are GenBank and the Protein Data Bank (PDB) [11, 35], (http://www.pdb.org). GenBank allows for automated submission but depends on a curator review before submission is complete. PDB, with the tool ADIT, [113, 114] has a fully-automated annotation and deposition process. However, there are still instances where a curator needs to intervene. These databases have created new and interesting problems within database research.

Biological and evolutionary (phylogenetic) databases tend to have specific needs that are not normally addressed with common databases. Some excellent examples of phylogenetic and evolutionary data repositories can be found at [29, 34, 47, 65, 81, 94]. First, these databases tend to be very dynamic. Data is constantly added to these databases. This process of adding data can take place through a number of methods. Some databases use a manual editing approach while others allow authors to submit data through World Wide Web forms. With the manually editing form, while most are still submitted through a Web interface, a curator needs to review the contents of the submission. The curator then can manipulates or annotates the data according to the need of the database. Examples of this type of database include GenBank and TreeBASE. The Protein Data Bank uses a fully automated tool for deposition, called "ADIT". ADIT allows the researching scientist submitting results to submit through the World Wide Web. ADIT takes the submission and proceeds to process the submission, ensures the submission conforms to the standard the Protein Data Bank uses. This submission can also be annotated manually as needed [113, 114]. Moreover, knowledge about this data is also constantly being discovered. This knowledge can have great affects on the importance placed on specific types of data within a database. It can affect everything from the data itself to the schema. Also, this affects legacy data within the database. It can have consequences with analysis of the data and interoperability among the databases.

Looking at phylogenetic data as a model for applying data cleaning to biological data, there are a number of issues within this field that are both interesting and unique to the phylogenetic field. Phylogenetic data is any data related to the evolutionary knowledge of a particular species. Phylogenetics is an area of study dedicated to researching the evolutionary history and relationships of species. The primary goal of phylogenetics is to understand how a species evolved. Understanding the evolution of an organism or species is a complex task that requires biologists to understand both the molecular composition of a species as well as the effects of the species' environment on its development. The molecular composition of a species describes the species' genetic structure as well as the arrangement of the species' biological system. The effect of the species' environment considers all outside factors that effect the species' development. This can include information about its place in the food chain, its mating practices, if it is in a host-parasite relationship with other species as well as many other factors. These environmental effects as well as the species biological composition determine the possibility for the perpetuation of the species [79].

Moreover, through trying to understand the evolutionary history of a species, biologists can uncover its relationship to other species. It is theorized that all life comes from one initial species. Phylogenetic research allows biologists to try to trace the development of current species to this one original species by organizing phylogenetic trees.

Phylogenetic trees are tree structures that model the evolutionary relationships among species. This tree's leaf nodes represent currently existing species. Internal nodes represent species from which the species in the leaf nodes descend. These trees are developed based on the various characteristics that species have. Currently, the most popular methods of developing phylogenetic trees, called phylogenetic reconstruction, compare the genetic structures of a set of species and decide how closely they are related. The species most closely related become siblings or closely related through a common ancestor. In this method, usually a node will have a descendent if a significant change occurs within the genetic structure such that a new species or a set of species results. However, phylogenetic reconstruction methods are not limited to this [79].

Understanding phylogenetic information about a species can be very beneficial in many other areas of research. By understanding the phylogenetic information about a species, scientists can understand the development of the species. For example, understanding the evolutionary development of whales can explain why whales are classified as mammals as oppose to fish. Furthermore, it can help define the classification system for species, thereby giving scientists criteria for classifying previously unknown species. Moreover, it can help predict the future evolutions of the species or explain why two species, with a common ancestor, branched from that ancestor to create two new species. It can also help track the evolution of a species. For example, the mutation of viruses can be tracked this way. Since viruses have distinct genetic compositions, and viruses work through mutations, various strains of the same virus can be analyzed using phylogenetic analysis of their genetic composition. Phylogenetic trees can be created from this information; possibly showing which strain of the virus is older. This could then give scientists indications as to how the virus was transmitted from one host to another, showing contagion patterns [79].

With the developmental information about the species as well as the information generated by the phylogenetic reconstructions, the data surrounding phylogenetic studies becomes complex and difficult to manage. This complex data needs various types of tool that will allow users to effectively manipulate the data. Therefore, it becomes imperative that this data is as "clean" as possible. Issues concerning cleaning this data become complex since, not only is the database working with data in traditional record formats, but also the database contains data that models the structures of the phylogenetic relationships.

Moreover, with the current nature of biological databases, the database will need to interact with other database. Therefore, the data needs to be clean so that integrating new submission data as well as data from other sources is possible. To clean text errors, issues develop about how to clean text since text can represent biological sequences or names of species. Also, the data of primary interest is the phylogenetic trees. The issues concerning how to clean phylogenetic trees become important. For example, there is a need to ensure that every species that is a member of the phylogenetic tree is correct since this can indicate input errors.

The issue of whether this tree exists already within the database also becomes of interest. This can indicate a duplicate record or related information a user might be interested in. Knowing what trees are similar within the database is extremely important. This can also indicate duplicate records. If other aspects of the phylogenetic data are included within this similarity test, there are informative comparisons that can be made about the algorithms used to create the trees. It can also indicate duplications or errors in research. For example, if two researchers use the same method to develop similar trees yet the trees are very different, this could indicate an error in submission of the tree, an error in the tools the researcher used or an error within the algorithm.

When analyzing biological databases generally, a number of problems can be identified concerning data. First, most data in these databases have been accumulated. Commonly, no strict database rules have been applied to it, causing erratic schemas. These schemas can be difficult to understand by visual inspection and even more difficult to design automated tools around for knowledge discovery. Also, as many of these databases age, data uniformity becomes an issue. In the biological fields, innovation and knowledge about the biological systems being studied are growing everyday adding to the knowledge about the data. Therefore, more aspects of the data being stored today are known then they were even five years ago. This causes inconsistencies within the data format. The Protein Data Bank (PDB) illustrates this issue very well. PDB is currently dedicating a lot of their resources to its "data uniformity project" [113, 114]. The data within the PDB is representative of over thirty years of research within microbiology and protein research. Necessarily, knowledge has increased during this time period, causing the amount and type of information stored to change over the years. For example, knowledge about the atomic structure of proteins has changed during this time period. This has necessitated that the data schema respond to such needs. However, previous data has been maintained in old formats. Also, there are nomenclature issues, where various atoms are misnamed. The old formats need to be reconciled as well as the nomenclature issues to make the database uniform [113, 114]. Besides causing data inconsistencies, it also causes the greater issue of how to update this data. Also, since the new data has more aspects, the schema for this data tend to be different than the earlier data.

These "dirty data" occurrences also cause other issues. For example, since there are multiple formats, and biological data is extremely complex, the problem of synonymy arises. In biological research it is very easy for researchers to consider a new species, disease or protein to be new while in reality, it is just a minor mutation of an already discovered biological entity. However, since the data sets are so large and the data management so difficult, it is extremely difficult to detect these problems.

Two examples of these "noisy data" problems can be seen in the phylogenetic tree database TreeBASE. TreeBASE, which makes its data public through the matrix files for their drawing tools, has a number of inconsistencies. TreeBASE, like many other biological databases, hosts many files that have legacy data. As knowledge about the data increased, formats were changed. For example, the representation for the genetic codes differs from file to file, usually depending upon when the data was researched. Representation of gaps within the genetic code is also file dependent. Also, representations for the trees themselves change slightly from file to file. While all trees are in Netwick Notation, the metadata surrounding the trees change format. Therefore, it can be difficult to process the trees.

The issues of data integration and interoperability are also important within this field. As more and more biological databases develop, the need to be able to port information seamlessly from one tool to another becomes important. Even among data banks where the data stored is nearly identical, many times those databases have specialized tools that no other databank has. Ideally, it would be advantageous to allow biologists to use whatever tool he or she wants to analyze whatever data he or she wants.

Therefore, given a biological database and specifically a phylogenetic database, this framework should ultimately reduce if not eliminate "dirty" data and schemas from the database. It should return to the user the clean database. To do this, first the biological database must be analyzed. The issues concerning the schema and the issues concerning the data need to be separated. Next, various cleaning operations will be performed upon the schema and data. These cleaning operations include a method for modifying schema (MAP), detecting duplicates or errors (MATCH,

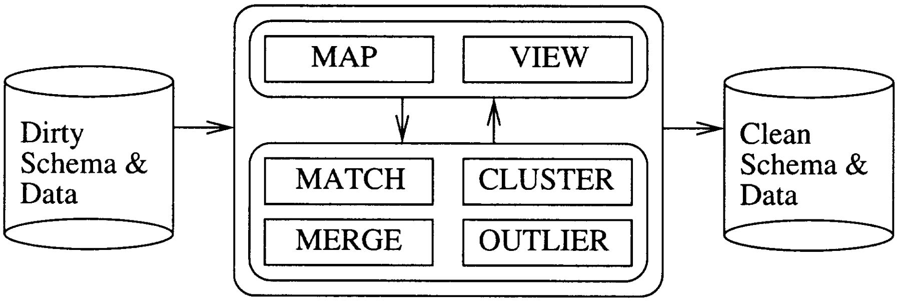


Figure 3.1 A framework for biological data cleaning.

CLUSTER, MERGE and OUTLIER), and viewing any section of the database needed (VIEW) [32, 33]. Once the data and schema are cleaned, the new data is staged for re-integration into the database. This allows for any integration of the cleaned data to be performed as well as any other operations need before the data is re-integrated. Finally, the data and schema is re-integrated, resulting in the cleaned database. This framework is demonstrated in Figure 3.1.

While this framework is conceptual, there are a few requirements needed of this framework so that it can perform effectively on biological databases. Primarily, it must be extensible. As it will be further explored in the data cleaning section, many cleaning frameworks are not extensible. If the framework is not extensible, then it cannot be flexible enough to support the various operations that would be needed to clean biological data. Biological data tends to be complex, incorporating structural data, keyword based data, metadata as well as other forms of data. Moreover, with the multitude of biological databases available on the Web, a framework must be able to consider the different types of data and different categories of data these databases can hold. Some databases hold nucleotide information while other protein while other publications about biological data. An extensible framework is imperative so that it can respond to these needs.

CHAPTER 4

DATA CLEANING

As mentioned previously, data cleaning methods address the issues concerning the quality of data [23, 24]. 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" [23, 24].

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 [23, 24].

Next, there is duplicate detection. 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 [82].

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 synchronized with other databases. With interoperability being facilitated through the World Wide Web, many databases tend to need to synchronize with other databases. This process of synchronization 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 [82].

Data cleaning problems have been observed throughout the history of databases, with the first documented cases stemming from the United States Census information during the 1950's. Since then, there have been many developments within the data-cleaning field. Commonly, a data cleaning system should fulfill 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 [82].

4.1 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 et al., [12]. 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 favor of speed. Bitton's method was to modify 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 [12].

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 neighborhood approach to detecting duplicates. In the sorted neighborhood 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 [43, 44, 62].

While the sorted neighborhood 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 et al., [43, 44] developed a modified sorted neighborhood 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 [43, 44].

There is also an incremental method to cleaning databases based on this multiple-sort sorted neighborhood 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 neighborhood 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 neighborhood 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 cluster it is most similar to. The representative is then modified to reflect the new record [43, 44].

4.2 Knowledge-based Methods

Besides using methods with various applications of sorting, there are a number of machine learning methods for performing data cleaning [20, 62]. 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 Lup Low et al., [62]. Intelliclean first "scrubs" the data for irregularities that can be detected easily. For example, it standardizes 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 standardized 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 [62].

While many data cleaning tools are automated, there are also other tools that require user interaction to perform cleaning. Potter's Wheel developed by Raman and Hellerstein [83] at the University of Berkeley in California is such a system. With interactive tools, data cleaning can be more specified towards the user's specific 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 [83].

Another method, proposed by Caruso et al., [17] 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 then trained on the selected data to obtain the rules for detecting duplication. It is then applied to the entire data set [17].

4.3 The Extraction, Translation and Loading 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 [82]. ETL's primary tools are data flow graphs, which tracks the transformation of the dirty data into the cleaned data [82]. The ARKTOS system, a data cleaning system that implements the ETL structure is an example of a system that uses this structure [100].

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. 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 standardized 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 [32, 33].

4.4 Declarative Data Cleaning

To address some of the problems identified in biological databases as well as those raised concerning some of the data cleaning methods, the declarative data cleaning method can be used. The data cleaning method proposed by Galahardas et al. in "Declarative Data Cleaning: Language, Model and Algorithms" [32, 33] defines a method, or more precisely a set of operations that are essential to any data cleaning tool. These operations are specified in the following section. They are specified in a BNF-styled notation where non-terminal symbols are contained within angled brackets '<' and '>', alternative productions are indicated with '|', terminal symbols are indicated in capital letters, a production enclosed within brackets '[ ]' indicates that production is optional and a production enclosed within curly braces '{ }' can be repeated one or more times [32, 33].

4.4.1 The Map Operator

Mapping generally is defined as an operation that takes a tuple from one relation and transforms it into a tuple for another relation. For example, given two databases for personal information, in one database, the name of a person could be represented as (lastName, firstName, middleInitial) while in a second database, this information could be represented as (lName, fName, mIn). A mapping operator would specify how to transform the first schema into the second schema. The mapping operator can be very useful for the creation of new relations as well as transforming data from alternative sources into a format the database can process [32, 33].

The following includes the syntax for the mapping operator from [32, 33]. Also included are the syntaxes for the let clause and the output clause, which are a part of the syntax for the mapping clause as well as some of the operators that are introduced in this section.

Syntax for Mapping Operator:

```
<mapping-operator>  : create mapping <operation-name>
                      from <predicate-name> [<alias-variable>]
                      [let <let clause>]
                      [where <where-clause>]
                      <output-clause>
```

The syntax for the mapping clause is very similar to that of a SQL statement. The first line "create mapping <operation-name>" specifies what operation is to be performed as well as an identifier for that operation. The "from" clause specifies the standard SQL FROM clause. It specifies the proper input predicate as well as the alias variable associated with the input predicate to be used throughout the rest of the operator statement [32, 33].

The "let clause", which is used in a few of the AJAX operations, creates a predicate key for the operation by calling to atomic functions that generate the key. The syntax for the let clause is given below. The let clause's syntax includes functional assignments (<functional-assignment>), control statements (<if-then-else-expression>), SQL-like statements (<sfw-expression>), and exception handling (throw <exception-name>). This gives the mapping powerful control over the data. The functional assignments are atomic statement that allows a predicate to be assigned a value. The control statements allow for selection of specific statements as well as exception handling. Finally, the SQL-like statements allow for further refinement of the data [32, 33].

The "where clause" is similar to the SQL WHERE clause, in which it acts similar to a filter for the operation. Finally, the "output clause" organizes the data wanted from the mapping operation (or any other operation using the clause) into the desired schema. As shown in the below table, the "output clause" is a modified SQL statement. The "select-into" statement stipulates the schema of the target relation as well as the constraints on the relations [32, 33].

Syntax for "let clause"

```
<let-clause>                  : <assignment-statement>, [{<assignment-statement>}...]
<assignment-statement>        : <predicate-name> = <functional-assignment>
<functional-assignment>       : <functional-expression> | <if-then-else-expression> | <sfw-expression>
<if-then-else-expression>     : if <condition> then <then-else-expression> [else < then-else-expression>]
<then-else-expression>        : throw <exception-name> | <functional-assignment>
<functional-expression>       : <function-name> (<arg-expression> [{,<arg-expression>}... ])
<arg-expression>              : <constant>|<domain-variable>| <predicate-name>| <functional-expression>
<sfw-expression>              : select<sql-project-clause>
                              from <sql-from-clause>
                              where <sql-where-clause>
```

Syntax for "output clause"

```
<output-clause>          : <select-into-clause> [{<select-into-clause>}...]
<select-into-clause>     : <sql select-into> [{constraint <constraint-clause>}...]
<sql-select-into>        : select <sql-project-clause> into <predicate-name>
<constraint-clause>      : unique <att-name>[{<att-name>}... ] | not null
                           <att-name>[{<att-name>}...]
                           | foreign key <att-name>
                           [{<att-name>}...] references
                           <predicate-name> (<att-name>
                           [{<att-name>}... ] )
                           | check <where-clause>
```

4.4.2 The View Operator

The view operation corresponds to an SQL query with modifications that can help with exception handling and integrity constraints [32, 33].

Syntax for View Operator

```
<view-operator>  : create view <operation-name>
                   from <predicate-name> [<alias-variable>]
                   [{<predicate-name> <alias-variable>... }]
                   [where <where-clause>]
                   <select-into-clause>
```

The syntax of the view operator is the syntax of any SQL statement that can be used to extract information from the database. The "create view" statement creates the predicate that will hold the information obtained from the SQL statement in the <operation-name> predicate [32, 33].

4.4.3 The Match Operator

The matching operator computes the distance 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. In a traditional database, matching can be computed through joining two relations and using an arbitrary distance measure [32, 33].

Syntax for the Match Operator

```
<matching-operator>  : create matching <operation-name>
                       from <predicate-name> [+] <alias-variable>
                       [{<predicate-name> [+] <alias-variable>...}]
                       [let <let-clause>]
                       [where <where-clause>]
                       into <predicate-name>
```

The match operator's syntax is very similar to that of both the mapping operator's syntax and the view operator's syntax. The match operator's syntax is primarily that of an SQL-statement. However, to perform the more advanced matching operations, the "let clause" is used. Also, the [+] in the "from clause" indicates that the predicates that do not have a match should be returned as well [32, 33].

4.4.4 The Cluster Operator

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. Generally, there are two methods from which clustering can be performed. Clustering can be performed based on the data value within a specific attribute. This is similar to the "GROUP BY" query in SQL. The second method is to define a distance measure for a specific attribute, defined as the clustering key, and cluster by that attribute. Since this is distance based, many standard methods in data mining for clustering can be applied using the second method [32, 33].

Syntax for the Cluster Operator

```
<cluster-operator>  : create clustering <operation-name>
                      from <predicate-name> [+] <alias-variable>
                      by method <method-name>
                      [with parameters <parameter-name>
                      [{<parameter-name>...}]]
                      into <predicate-name>
```

The cluster operation's syntax uses two new constructs, the "by method clause" and the "with parameters clause". The "by method" clause specifies the method that is to be used to cluster the data while the "with parameters" clause specifies the parameters for that clause [32, 33].

4.4.5 The Merge Operator

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 defining a tuple is returned as output. This relation groups and collapse the set of elements within the input relation based on some matching criteria [32, 33]. *Syntax for the Merge Operator*

```
<merge-operator>   : create merge <operation-name>
                     using <predicate-name> [<alias-variable>]
                     let <let-clause>
                     [where <where-clause>]
                     <select-into-clause>
```

The merge operator's syntax uses similar clauses that have been previous explained. However, instead of selecting predicates through the "from clause", there is now a "using clause". In other operators, the "from clause" allows the operator's "let clause" to be specified with respect the predicates from the "from clause". In the merge operator, the "using clause" allows for the each cluster to be evaluated instead of each predicate [32, 33].

These operations were then instantiated into a system called AJAX. AJAX, developed by Galhardas et al., [32, 33], 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 [32, 33]. 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 [32, 33].

CHAPTER 5

THE BIO-AJAX FRAMEWORK FOR BIOLOGICAL DATA CLEANING

The BIO-AJAX framework is a proposed data-cleaning toolkit for biological databases that is designed to clean both schema level and data level data quality problems. It will use the conceptual operations presented by Galahardas et al., [32, 33], presented in Chapter 4, that ultimately developed the AJAX system as well as adding a new operation to process outlier data from clustering. The BIO-AJAX framework will preserve 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 modeled 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. 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.

5.1 The BIO-AJAX Toolkit Operators

5.1.1 The MAP Operator

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.

5.1.2 Mapping 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 standardized. 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.

Example 1: One common procedure within phylogenetic databases is to extract trees for a knowledge discovery tool, such as a comparison tool or a drawing tool. Since data files within these databases tend to be large, it is extremely useful to extract only the data needed for cleaning. Moreover, sometimes the trees within these files can be in different formats. The mapping operation can be used to extract the relevant data, such as the tree structure, from the dirty data table for further processing. Based on the grammar in the previous section the following query would extract a key from the data using an atomic function as well as the tree:

```
CREATE MAPPING ExtractPhyloTree
FROM dirtyPhyloData dph
LET tree = extractPhyloTree(dph.data), key = generatePhyloKey(dph.data)
WHERE PhyloData.tree < > null
{SELECT dph.key AS phyloKey, dph.tree AS Tree
INTO DirtyPhyloTreeData
CONSTRAINT NOT NULL phyloKey}
```

The extractPhyloTree function can extract the tree from the data. This extraction may require extracting the tree from files with other information. This also may require that the tree be formatted separately from its original format so that it can be mapped according to the new schema.

5.1.3 The VIEW Operator

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.

Example 2. The following SQL command displays the phylogenetic information exacted by Example 1.

```
CREATE VIEW viewTrees
FROM DirtyPhyloData d1, DirtyPhyloData d2
WHERE d1.phyloKey = d2.phyloKey
{SELECT d1.phyloKey AS key, d1.tree AS tree INTO Trees
CONSTRAINT NOT NULL tree}
```

5.1.4 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 sapiens, 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 standardizing 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 standardizing nomenclature, comparisons and other knowledge extraction tasks can be performed much more effectively and efficiently.

Besides nomenclature, matching can also be 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 [60].

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 standardize 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 in 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 analyzing 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.

5.1.5 Matching and Phylogenetic Data

A number of similarity and distance measures for phylogenetic trees have been proposed. Various algorithms for tree matching [107] and for tree reconstruction have been studied. Different theories, when applied to finding the phylogenetic relationship of the same set of species, often result in different phylogenetic trees. To determine how much two theories have in common is a fundamental problem in computational biology and in phyloinformatics.

The most comprehensive algorithmic and software tool in this field is perhaps the COMPONENT package (`http://taxonomy.zoology.gla.ac.uk/rod/cpw.html`) developed by Page at University of Glasgow. It provides several ways of finding the consensus of two phylogenetic trees. The first is to see whether they have similar "quartets", which are based on adjacency relationships among all possible subsets of four leaf species. The similarity is then computed as the proportion of quartets that are shared in the two trees.

Partition distance treats each phylogenetic tree as unrooted and analyzes the partitions of species resulting from removing one edge at a time. By removing an edge in a tree, one is able to partition that tree. The difference between two trees is defined as the number of edges for which there is no equivalent (in the sense of creating the same partitions) edge in the other tree.

The maximum agreement subtree between two phylogenetic trees T_1 and T_2 is a substructure of the two trees on which the two trees are the same. Commonly such a subtree will have fewer leaves than either T_1 or T_2. By contrast, a consensus tree has the same number of leaves as the original trees T_1 and T_2, assuming those trees have the same set of species. Consensus trees should be used gingerly, however, because a consensus tree is not a phylogeny unless the two trees are isomorphic. Instead, consensus trees are a convenient way to summarize the agreement between two or more trees. Consensus trees can be formed from cluster methods (strict, majority rule, semi-strict, or Nelson) or by intersection methods (Adams).

The last dissimilarity measure implemented in COMPONENT is the nearest neighbor interchange (NNI) distance. Given two unrooted unordered trees T_1 and

T_2 with the same set of labeled leaves, their NNI distance is the number of NNI operations needed to transform T_1 to T_2. Finding the NNI distance between two trees is NP-hard. Brown and Day developed several approximation algorithms to calculate the distance, which are implemented in COMPONENT.

The following example demonstrates how matching can be performed using the research from [107] that was eventually implemented into the ATreeGrep and TreeRank [89, 92, 106] algorithms for searching unordered trees [92]. ATreeGrep and TreeRank measure similarity of structures through an approximate nearest neighbor distance. In this example, two trees are compared and the degree of similarity can be controlled through the third parameter of MAXDIST in the WHERE clause.

Example 3: Matching in phylogenetic data using ATreeGrep

```
CREATE MATCHING MatchDirtyPhyloTrees
FROM DirtyPhyloTreeData d1, DirtyPhyloTreeData d2
LET distance = aTreeGrepDiff(d1.tree, d2.tree)
WHERE distance < MAXDIST (d1.name, d2.nam2, 0)
INTO MatchTrees
```

5.1.6 The CLUSTER Operation

The clustering operation can perform similar duplicate detection functionalities that the matching operator can perform. Conceptually, the cluster operation organizes 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.

5.1.7 Clustering and Phylogenetic Data

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 analyzed 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 [96].

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 [96].

Example 4: This example clusters phylogenetic trees. The clustering operation is very interesting for this data set. In this example, trees are clusters based on similarity. This can be highly useful for comparing phylogenetic trees and their methods of construction.

```
CREATE CLUSTERING clusterTreesbyStockhamMethod
FROM MatchTrees
BY METHOD StockhamMethod
INTO clusterTrees
```

5.1.8 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 analyze 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 clustered, 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.

Syntax for the Outlier Operator

```
<outlier-mapping-operator>   : create <outlier-mapping-name>
                               using <predicate-name> [<alias-variable>]
                               let <let-clause>
                               where {cluster-size = 1} [<where-clause>]
                               <select-into-clause>
```

5.1.9 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 [88]. 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 form one tree. This one tree preserves as best as possible the evolutionary relationships between the species within all of the trees [88, 90].

Since scientists studying phylogenetics theorize 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 [88, 90].

Example 5: This example merges similar data within a phylogenetic database. Merging phylogenetic data has a number of uses. First, if exact matches are found and it is a duplication error, the duplication can be eliminated. Second, if the database wants to condense similar trees, merging can perform this task. Finally, merging can also facilitate the formation of supertrees trees.

```
CREATE MERGING MergeTrees
USING clusterTrees ct
LET tree = getTreeCluster(DirtyPhyloTrees(ct).tree) key = generateKey()
{SELECT key AS phyloKey, tree AS PhyloTree INTO Tree}
```

5.1.10 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.

Below is the syntax for the CLASSIFY operator. The syntax for this operator is similar to that of the MATCH operator, since their functions are similar. However, classify helps to identify a data item as oppose to comparing it against other data items within the repository to detect similarity.

```
<classify-operator>   : create classification <operation-name>
                        from <predicate-name> [+] <alias-variable>
                        [{<predicate-name> [+] <alias-variable>...}]
                        [let <let-clause>]
                        [where <where-clause>]
                        into <predicate-name>
```

Concerning phylogenetic data, there are many instances where classification may be needed. Concerning nomenclature problems, classification can offer the ability to standardize 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 analyzing 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. Example 6 illustrates this problem.

Example 6: Classifying a phylogenetic tree that is missing reconstracution method implementation.

```
CREATE CLASSIFCATION classificationTreeReconstructionParsimony
FROM MatchTrees
BY METHOD ParsimonyClassifier
INTO classParsimonyTrees
```

CHAPTER 6

DATA INTEGRATION AND BIOLOGICAL DATABASES

6.1 Introduction

One important data integrity issue that all biological databases need to address is the data integration problem. Traditionally, data integration usually applies primarily to the problem of combining data from different sources, providing the user with a unified view of this data [59]. Data integration offers biologists many opportunities to exploit the data housed in biological databases more fully. Through applying various knowledge discovery techniques, it offers biological databases the ability to interact with various levels of data and give users a straightforward interface to the immense amount of data biological databases manage.

Biological database integration is a popularly researched topic that has resulted in a number of innovative methods for the interaction of data among databases, particularly those accessible through the Web [1]. Its applications can include anything from integrating databases, so various knowledge discovery tools can be applied to data distributed to various databases, to providing users with a simple interface. However, traditional concepts concerning data integration can limit the integration possibilities for biological databases. With biological databases, the integration problem becomes more difficult than creating a seamless view of a set of data for a user. Integration must be considered not only as a problem affecting interacting databases but also as a problem that affects multiple areas of the database. These areas include when submitting new or modifying data to the database, having data within a large repository interact with each other and having separate databases present a unified view of data from multiple database projects to a user. Ideally, most biological databases need to address data integration on at least three levels. These levels include:

- integrating submission data;
- integrating data currently within the database;
- integrating data with external data repository resources.

Integrating data based on these levels offers biological databases the ability to house data more effectively while giving users more control when researching. It allows for scientists to gather their results in one location and standardize the format of their results. It gives curators of these databases a tool that frees them from creating one, centralized database for their data. Finally, it also allows independently developed databases to interact with each other, allowing users to take advantage of the abilities of using knowledge discovery tools from multiple databases in one interface.

6.2 Data Integration

As mentioned in the previous chapters, many data cleaning problems arise from how data in different schemas interact with each other or map to a common view of the data. This affects the retrieval recall and precision of a data repository, its ability to interact with other data repositories and the repository's ability to incorporate or modify the model of the data. Whether the cleaning problem is one of these problems or some other problem where ultimately, data from one set must be mapped to another, it can be viewed as a data integration problem.

Data integration is a well-known problem within both the fields of database management and bioinformatics. Commonly, the data integration problem is seen as the problem of combining data residing in different data sources into one seamless view for the user [59]. This frees the user to be concerned only with the query rather than locating sources for the data they are interested in or collating data from multiple sources to get a complete view of the data.

A generic data integration system typically consists of three parts: the global schema, the source schema and the mapping between the global schema and the source schema [59]. The global schema refers to the combination of the data residing in or originating from various sources. The source schema refers to organization of the data obtained from the source data sets. The mapping describes how the data in the source schema participates in the global schema. For the integration system to be acceptable, the integration must satisfy the constraints imposed upon it by the global schema while also satisfying the mapping between the global schema and the source data sets.

The two most common data integration approaches are the "local as view" (LAV) and "global as view" (GAV) methods. Both of these approaches define methods for creating the mapping between the global schema and the source data sets. In the LAV approach, the global schema is specified independently of the source data sets. The source data sets are seen as views over the global schema. The GAV approach specifies the global schema in terms of the source data sets. Each element of the global schema is seen as a view over data within the source data sets [59].

Both LAV and GAV have problems associated with them. In the LAV approach, query processing is difficult. Since the only information known about the data in the global schema is through the views of the sources, which are limited, the data can be incomplete. While GAV does not have this problem, adding new sources or new data to the global schema can become difficult. To help maximize the benefits of the integration, some databases use a combination of these approaches, referred to as GLAV, to create the integration [16, 59]. Recent efforts within the data integration communities have looked mainly at continuing efforts concerning schema mediation languages, query answering algorithms, query optimization, query execution and creating applications using data integration techniques [36, 37, 38].

6.3 Biological Data Integration

Work in both general data integration and data integration in bioinformatics data sets has been extensive. For more traditional data sets, one important data integration system is BibFinder [75, 76]. BibFinder is a citation index for computer science articles which integrates citation searches from the collection of Computer Science Bibliographies (CSB), DBLP, Network Bibliography, ACM Digital Library, ACM Guide, ScienceDirect, IEEE Xplore, CiteSeer, and Google. BibFinder is a test-bed for the research done within the Havasu Project [75] and is innovative in that it addresses many complex issue within data integration. The Havasu Project works developing automated methods for executing a query over multiple, heterogeneous database simultaneously, accounting for the cost and optimization issues for such a query. The research from the Havasu Project has also been extended into BioHavasu, a mediator for biomedical genomic data. However, this mediator currently requires that the use knows the gene accession number to use the system.

Another important integration project is a component of the Foundations of Data Warehouse Quality Espirit Project (`http://www.dbnet.ece.ntua.gr/~dwq/`) [101]. This project is compromised of a number of research centers and universities in Europe to work at developing a basis for developing advanced databases facilities, such as deeper models, more complex and interesting indices as well as semantic services for data warehouses [101]. Within this project, data integration techniques have been used to help further improve data quality, mainly through integrating original sources into the warehouse.

The University of Washington the Database Group is currently working on a number of projects regarding integration. For example, the Piazza project investigates integrating peer-to-peer resources [38] using semantic mappings. It employs both LAV and GAV methods, using XML and its applications within Semantic Web Research as a platform to do such integration practices. Piazza is an extension of the Sagres system, which looks at data sharing among invisible and ubiquitous computing devices [48]. Another significant integration system developed the Tukwila Data Integration System. This system offers the state of the art in network query processing. Currently, this group is also working on BioMediator, which will help integrate genomic resources.

Besides the research at the University of Washington, a number of groups are looking at XML and Semantic Web Technologies for aiding in data integration. By nature, the Semantic Web Project is a data integration project [10]. In order to offer all of the functionalities the Semantic Web proposes, necessarily there will be a need for integration. Integration technologies for the semantic web include XML and the various communication layers associated with the Semantic Web language tower such as RDF and DAML+OIL. In [1], each group discusses how XML can be used effectively in biological data integration. Archard et al. give a brief overview of XML in bioinformatics.

Concerning conceptual biological data integration, there have also been a number of efforts. Biological data integration problems have been discussed in many papers, including [25, 57]. Many of the initial successful biological data integration projects primarily dealt with biomedical periodical data. A couple of successful projects based on this type of integration are BioKleisli [26], GeneScene [60, 61] and MedTextus [60, 120]. In [1, 55], the authors discuss XML as an excellent medium for interchanging biological data. From this basis of XML and Semantic Web, there are numerous integration projects. Within bioinformatics, some noteworthy projects include the Gen Ontology [13], the Tambis project [95], the BioFAST project and the Biological Integration System [56].

Within biological database, most freely available World Wide Web repositories have some level of integration. Archival databases tend to have multiple levels of integration. If the archival database is partnered with mirror sites, then the sites usually synchronized with each other regularly. Other databases that are not mirrors also connect in a less rigid manner, using Internet connections to query each others databases. For example, NBCI [8, 9, 98, 99, 112], uses "link out" to other prominent repositories so that the user can obtain more information on a particular subject.

Another interesting integration movement within this field is to use integration to unify the repositories. The Protein Data Bank (PDB) [11, 113, 114] uses various technologies to integrate new data into the repository as well as integrate legacy data and new data into one unified view of the repository. The Protein Information Resource (PIR) [116] also employs integration technologies to help search their various repositories. IProClass gives the user a unified search interface for the data found in from the PIR-PSD, SWISS-PROT, and TrEMBL databases [46]

Most of the integrated biological database systems address the concerns of providing the user with as much data as possible. However, integration can facilitate other areas of importance to biological database users. Biological databases have become more than data repositories but also analysis tools. Therefore, issues concerning both what data a biological database can recall as well as what tools can be applied to that data are of great importance.

Biological databases, like most databases, are subject to many challenges. However, due to the nature of the biological data, usually these challenges become complex thereby excluding the more traditional methods for solving the problems, complicating the integration problem. As mentioned before, commonly, the data integration problem within most databases concerns combining data from various sources to represent a unified view to the user of the data contained within those databases. Applying this definition to biological data, there are three specific areas data integration can then be applied to. These areas are integrating scientific data, integrating databases within one repository and integrating multiple repositories [42].

6.3.1 Integrating Scientific Data

The first area of integration, integrating the data scientists submit to biological databases, is always an interesting problem. Issues arise resulting from wanting to give scientists the freedom and the flexibility to specify their results descriptively while also controlling the submission so that integrating the data into the database is possible. If biologists are too limited during the submission process, key elements of their results can be expressed within the database improperly. Also, since knowledge within biology is dynamic, with new understanding of the data developing constantly, any information not included presently can affect results in the future. However, a number of problems can develop by not controlling the submissions into the database. First, by not controlling vocabulary, there will be very little consistency between entries. This can cause many problems, especially concerning retrieval of the data submitted as well as applying knowledge discover tools to the data.

Since the data is inconsistent, it becomes harder to retrieve data since two authors may use a different set of phrases to describe similar concepts. For example, two researchers enter nucleotide information about the species *Danio rerio.* One enters the data, classifying it as "Danio rerio" while the other enters the data in classifying it as "D. rerio x1", where *x1* would indicate the specific organism the sequence was obtained from. Without some consistency mechanism, whether it is a controlled vocabulary that restricts the user to a specific nomenclature format or a thesaurus that would reconcile the input, the later study would not be returned for anyone querying for *"Danio rerio nucleotide sequences".* Moreover, without any control mechanisms placed upon the submissions, it becomes very difficult to check for errors within the submissions. Submissions presented within controlled vocabularies can greatly affect the precision of the database. The synonymy and polysemy problems become difficult to control and query mechanisms become very complex.

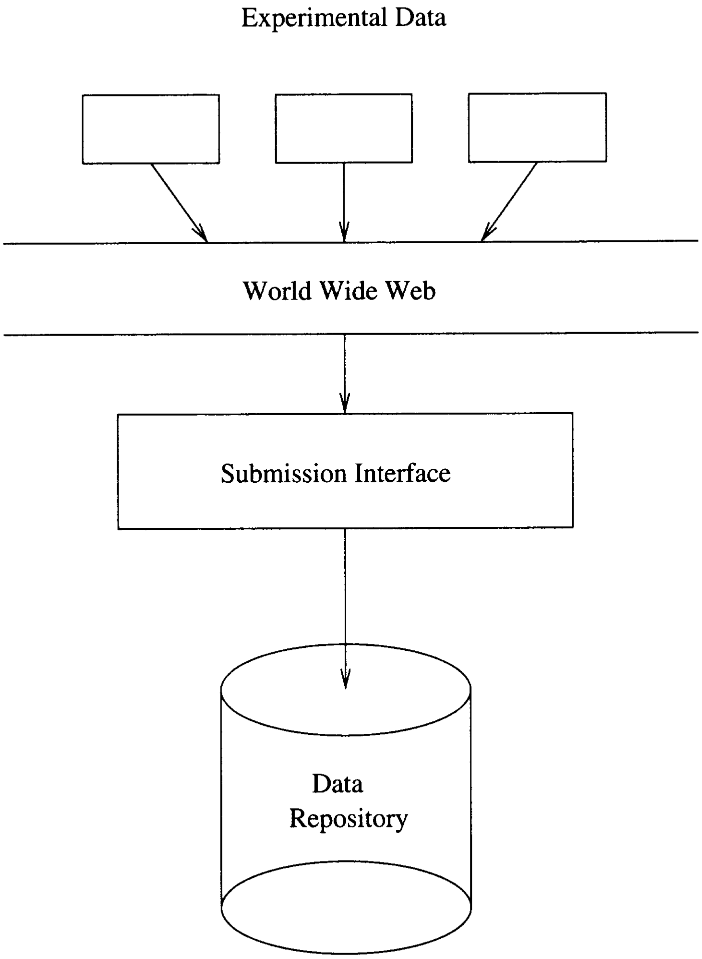


Figure 6.1 Integrating scientific data.

Figure 6.1 describes this type of integration. In Figure 6.1, a researcher would submit his or her data to the data repository. This data would most likely be submitted through the World Wide Web via some submission interface. Once the data is formatted through the interface, it is then stored within the database.

Currently, most biological databases use Web-based methods for accepting data submissions. These methods include a Web page interface that guides researchers through the submission process and email. The Web page interface is generally a Web form through which a researcher can submit data. If the researcher does not feel comfortable with this method, many database projects also offer the researcher the ability to email the data to them. If the data is complex, sometimes a curator will review the submission manually and then insert it into the database. If the Web form is not complex or if it has a strict controlled vocabulary, then it sometimes can be inserted automatically.

6.3.2 Integrating Databases Within One Repository

The second form of integration concerns integrating data within one data repository. For some of the larger biological database projects, multiple database management systems as well as data schemas are used to store the information planned for within the scope of the database. This type of database architecture is used for a number of reasons. Some databases house large amounts of legacy data. When there is a schema redesign for the database, the legacy data does not always conform to this new database schema. Therefore, the legacy schema can retain the legacy data and a new schema can be developed for the new data. An integrated schema can then be imposed upon this data. Also, since biological data is a complex data set, some DBMSs offer better functionalities for storing specific data than other database management systems.

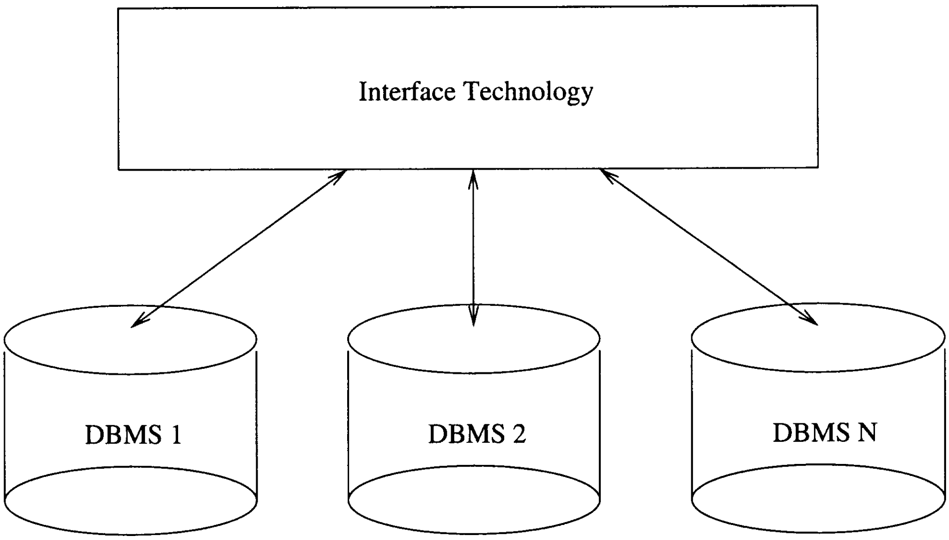


Figure 6.2 Integrating among data sets or DBMSs in a single repository.

Figure 6.2 demonstrates integrating within one repository. In this figure, there is a collection of data sets within one repository. The data sets are housed within DBMSs. These databases then interact through some integration technology. Possible integration technologies include some relational database tools, similar to the one used in PIR's iProClass [46]. Other possibilities include integrating the databases using Web technologies, as if each database was in its own separate repository. The user would then have a unified view of these databases.

6.3.3 Integrating Multiple Repositories

The third form of integration concerns integrating data from multiple biological database projects. Conceptually, this type of integration would allow a user to search for any topic. The interface would then return data stored in databases participating within the tool in a unified manner. Currently, there are a number of projects that partially implement this type of integration [26, 97]. For example, the EpoDB project (`http://www.cbil.upenn.edu/EpoDB/index.html`) uses similar techniques. The EpoDB is a database designed for analyzing the genes expressed during vertebrate

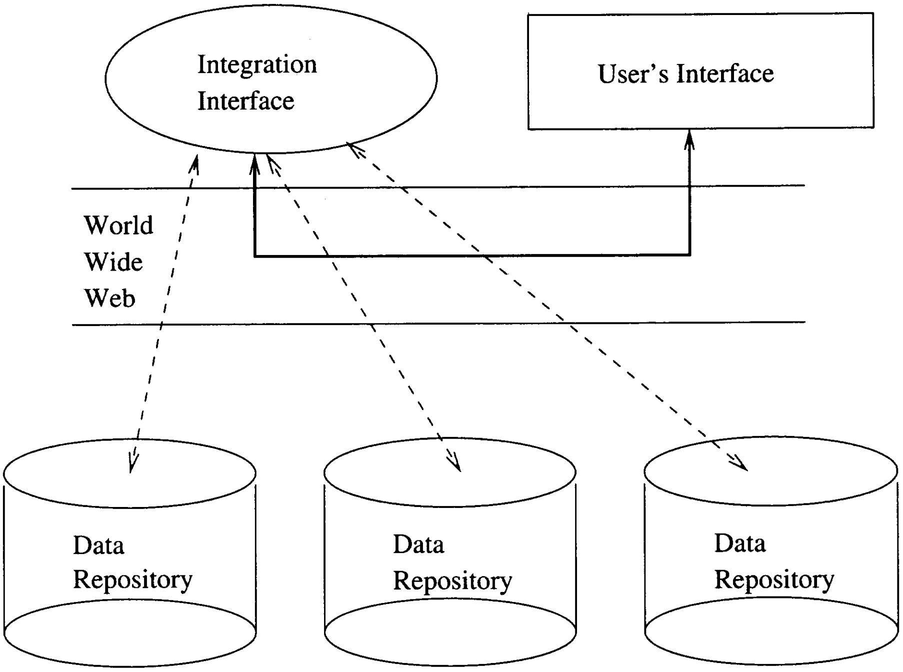


Figure 6.3 Integrating among data repositories through the World Wide Web.

erythropoiesis. This database creates a conceptual database using Sybase, extracting data from multiple sources including GenBank (`http://www.ncbi.nlm.nih.gov/`) and SWISS-PROT (`http://us.expasy.org/sprot/`) [97].

Figure 6.3 demonstrates this type of integration. This figure assumes that the user is interacting with the integration view of the database through a Web interface. However, this framework could be modified to represent any network medium. In this figure, both the user and the databases that take part in the integration are interacting with the integration interface. The user connects to the interface through the Web, represented through the dark line. The data repositories also connect to this integration interface. The integration interface is responsible for query processing. This connection is represented through the dashed line. The data repositories then interact with the integration interface to create a unified view for the user.

6.4 Data Integration and Data Cleaning

Through looking at data integration in terms of the aforementioned classifications, it can be seen how data integration can be used to help preserve data quality. Data integration can be used firstly to ensure submission data conforms to the schema currently stored data adopt. Moreover, when data repositories need to employ more than one DBMS to manage its data, data integration can help the repository present a seamless view to the user. This concept can be extended further by applying this model to multiple different schema that the repository is working with rather than just multiple DBMS. Finally, data integration plays a crucial part in cleaning data when multiple repositories interact with each other. Without integration, there would be excessive data duplication as well as a confusing representation of the data.

Extending this characterization, the mapping problem in the data cleaning framework can be viewed entirely as an integration problem. In data cleaning, mapping needs to be used for a number of occurrences. Essentially, at any point where the data needs to be transformed from one schema to another, this is a mapping operation. Data integration can be applied to facilitate this mapping, whether the case is mapping data from one schema to another or creating an environment for data in multiple schemas to interact with each other.

CHAPTER 7

THE BIO-AJAX TOOLKIT FOR TREEBASE

7.1 Introduction

Due to advances in biological research, information in biological data repositories has grown substantially. As with any database, these data repositories now must cope with a number of data quality issues that are inherent to very large databases. These data quality issues include synonymy, polysemy and data redundancy, to name a few. However, due to the complex and diverse nature of the data, the problem of improving the data quality is non-trivial. If the data quality is not maintained or improved, then for data extracted from the repositories by a third party or for data mining purposes, the applications and knowledge based on the inconsistent data will either fail or be skewed.

Data repositories are also expanding their functionalities, requiring more interactions among the data, thus creating more data quality problems. Currently, most data repositories are interrelating with each other, creating a number of integration problems such as reconciling both data and schemas. With biological research changing rapidly, creating more detailed information about biological processes, repositories must address the issue of how to integrate complex submission data with legacy data that follow different data models [63]. Many data repositories endeavor to reduce redundancy within their databases in favor of giving the user highly annotated consensus data. For example, the Protein Information Resource [116] strives to provide users with highly annotated information about the proteins stored within its databases. To provide this, its curators must manage the data through both manual and automated techniques. Data cleaning can offer methods for making the process more automated. Finally, the discoveries in biology affect not only the scientific community, but also other communities such as the business and finance communities. Therefore, there is a need for simple interfaces that give results in a concise, easy to understand style. All of the challenges can be interpreted as data quality problems and addressed through data cleaning and exploratory data mining [24].

The aim of this chapter is to use the extensible toolkit to address data quality issues through data cleaning within biological data repositories, specifically for phylogenetic data. As an example, BIO-AJAX is applied to solving the nomenclature problem in TreeBASE. TreeBASE is a phylogenetic and evolutionary information system, available at `http://www.treebase.org` [81] and accessible from GenBank. The system contains citation and experimental data for evolutionary studies. It can display the experimental data through dendrograms on the website. The user can then navigate these dendrograms, finding other dendrograms with similar taxa.

The nomenclature problem in TreeBASE is mainly concerned with the fact that nomenclature for evolutionary units is not entirely standardized. While Linnaean nomenclature (e.g. "Homo sapiens" or "Canis familiaris") is used pervasively within scientific communities, non-scientific communities use general vernacular terms such as "human" or "dog". This creates a number of limitations for evolutionary databases. First, it limits the community of users only to those familiar with the nomenclature. Second, it creates challenges concerning integration with other evolutionary biology resources. Finally, it creates inconsistencies in analyzing the data if there are non-standard interpretations of the nomenclature within the database.

In general, due to the type of data stored in TreeBASE, TreeBASE is unable to control the inconsistency problems within the database concerning the nomenclature. Ideally, all of the nomenclature should conform to a specific set of nomenclature, such as the Linnaean nomenclature, which TreeBASE strongly recommends within its submission guidelines. However, there are cases where the nomenclature needs to be slightly modified so that an experiment is properly modeled. For example, in an evolutionary study among organisms of the same species, each organism needs to be differentiated. Therefore, one common practice is to amend the taxon name, putting a suffix at the end of the taxon name unique to the organism. While this modification has great meaning to the study, it does create inconsistent data concerning the nomenclature and makes it difficult to have complete studies about all occurrences of a species within the database.

The benefits of this research can greatly improve many components of biological databases in general, and TreeBASE in particular. First, it can enhance information retrieval and knowledge discovery results within these databases. Any data mining performed on the database will reflect the content of the database rather than the inconsistencies within the database. Integration with other repositories can be conducted more efficiently. Finally, the groups of users viewing the data can be expanded since the scientific data can be mapped onto an uncomplicated and easy-to-understand representation of the data.

7.2 The BIO-AJAX Framework

The BIO-AJAX framework is a data cleaning toolkit for biological information systems that is designed to improve data quality at both data level and schema level [41]. It adopts and modifies the conceptual operations originally developed in the declarative framework AJAX [32] so that they specially meet biological data needs. Previously, most data cleaning methodologies were specific to the domain or greatly tied into the physical layers of the database. Their heuristics were highly incorporated into the implementation of the data cleaning tool. The tool AJAX, however, offered a way to encapsulate the primary operations of data cleaning to speak of the data cleaning techniques in an abstract manner. This gives flexibility to the system since the conceptual operations can be preserved while the instantiations of the operations can change as needed.

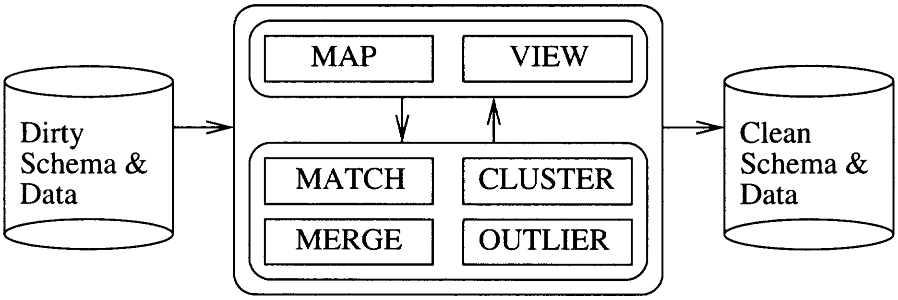


Figure 7.1 The BIO-AJAX Framework.

Figure 7.1 shows the BIO-AJAX framework consisting of six interrelated operators. These operators each have specific individual purposes but can, and in some cases need, to work with the results from other operations. These operators are:

- MAP: translates the data from one schema to another schema.
- VIEW: extracts portions of data for cleaning purposes.
- MATCH: detects duplicate or similar records within the database.
- MERGE: combines duplicate records or similar records into one record within the database.
- CLUSTER: identifies sets of relations within the data and organizes the data into those relations.
- CLASSIFY: analyzes a given data point, categorizing it according to various domain rules.

In general, there can be multiple extensions and implementations of these operators. For example, there are a number of algorithms that can perform matching in any data set, including phylogenetic trees. The operator CLASSIFY uses various classification algorithms to perform many tasks such as classifying a protein or helping to standardize metadata. The CLUSTER operator also performs operations associated with outlier detection in the data.

Implementing BIO-AJAX upon a data set first requires studying and detecting errors within the data set. This can involve using data mining algorithms catered to the data set to detect dirty data as well as discussions with curators and experts in the fields to understand observed problems. The next step is to identify algorithms that can effectively detect and clean the dirty data. These algorithms are tested on a small set of the database to gauge their effectiveness. If the algorithms are effective, they are then implemented onto the entire database and integrated into the cleaning tool. The tool then runs automatically, without any more curator interaction.

7.3 The BIO-AJAX Toolkit and TreeBASE

One of the data quality problems concerning phylogenetic data in general, and TreeBASE specifically, is the nomenclature problem. In evolutionary biology research, various nomenclature issues can arise. See the NCBI TaxBrowser Website where there are resources entirely dedicated to updating users on nomenclature changes [29]. While the curators at TreeBASE specify directions for nomenclature, many submissions do not follow these directions.

While currently there are many efforts within biological databases to start addressing errors and providing means for correction, none have been applied to phylogenetics. Most efforts involve looking at improving genomic and proteomic data. There have been many studies of problems within these databases [15, 31, 45, 70, 115]. Moreover, there are a couple of projects also addressing some cleaning issues. Such projects include AutoEditor, which looks to help correct genome annotation [31] and BIO-AJAX for TreeBASE [41].

Figure 7.2 illustrates this nomenclature problem. In this example there are three trees, all of which model evolutionary relationships among the species "Homo sapiens". In the case of these trees, each tree studies evolutionary relationships between specific organisms within the species. T_1 shows the evolutionary relationships

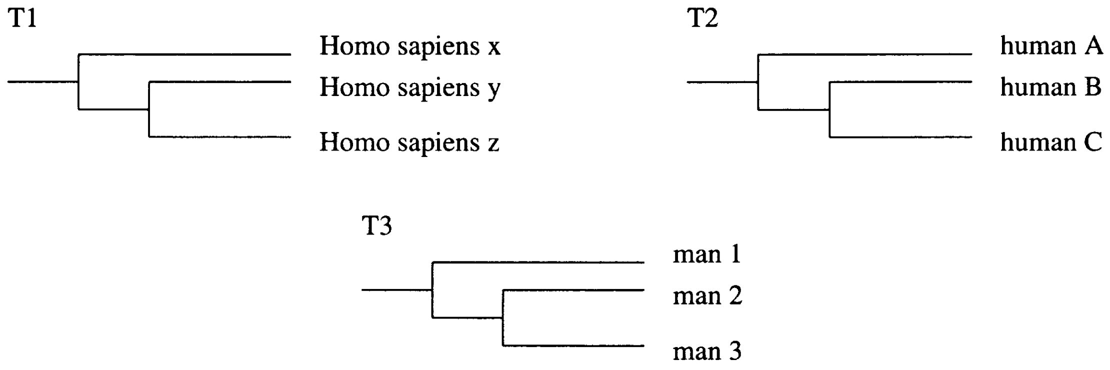


Figure 7.2 Example illustrating the nomenclature problem.

among three taxa "Homo sapiens x", "Homo sapiens y", and "Homo sapiens z". T_2 illustrates the relationships among three taxa "human A", "human B", and "human C". T_3 shows the evolutionary relationships among three taxa "man 1", "man 2", and "man 3". Now, consider tree T_1. The letters after the taxon name "Homo sapiens" indicate that a specific organism within the species or some other specificity is involved concerning the taxa within the tree. Therefore, this tree may not be generally about "Homo sapiens" but about a specific set of Homo sapiens organisms. In similar instances in TreeBASE, if we query for "Homo sapiens", most likely we will not get these three trees since the taxon "Homo sapiens" is not explicitly and exactly specified in the trees. (Only querying for "Homo sapiens x", for example, will allow us to get T_1.) Similarly, if we query for "human" or "man", we will not get the three trees even though all the three trees are related to "human", "man", or "Homo sapiens".

Thus, because of the inconsistency among the taxon names, we are unable to get a complete set of trees about the species "Homo sapiens". To solve the inconsistency and incompleteness problems, BIO-AJAX cleans the taxon names by implementing a layer between the user and the original database while not modifying the experimental data. Keeping the data intact is necessary since TreeBASE is an archival data repository where the original experimental data needs to be preserved.

Specifically, the operators of BIO-AJAX are instantiated to clean the nomenclature in the following manner (MAP and CLASSIFY are not relevant here and are omitted):

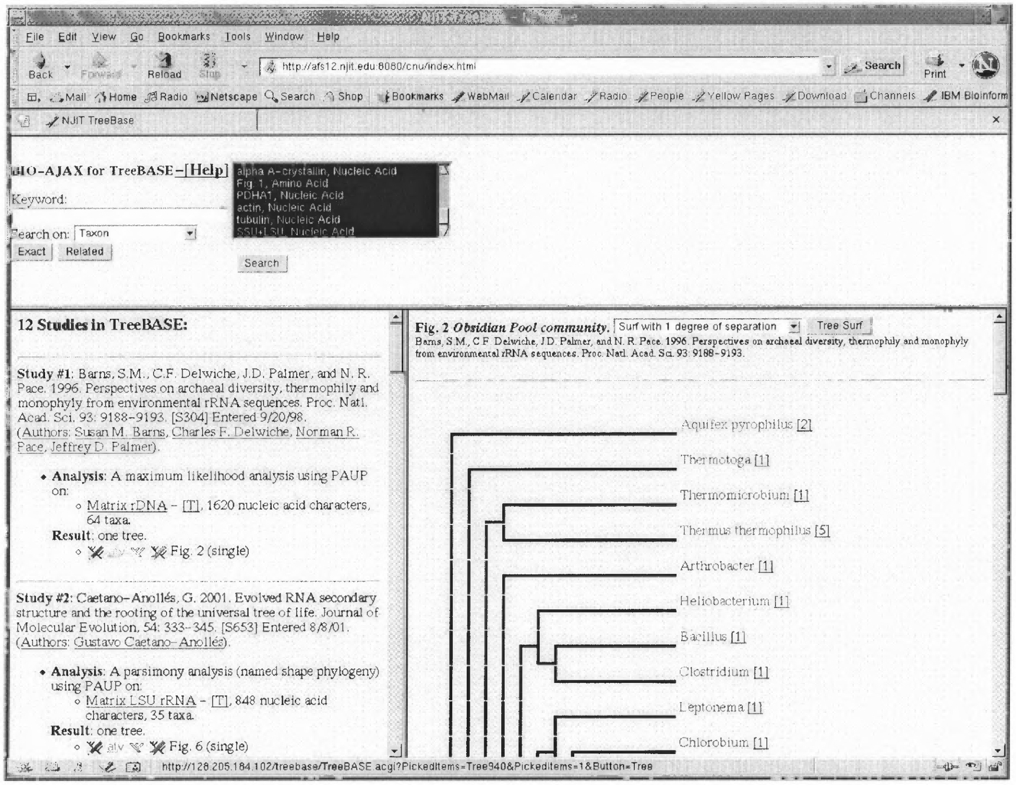
- VIEW: generates a list of prefixes for all taxa contained within TreeBASE.
- MATCH: compares the list of prefixes with the taxonomy entries in NCBI Taxonomy database. If an entry is found, obtain all nomenclature associated with the entry and index the names.
- MERGE: indexes the nomenclature so that any one of the prefixes, original nomenclature or nomenclature found in NCBI Taxonomy database can be used to query and obtain related trees.
- CLUSTER: uses hashing methods to group nomenclature and prefixes together that would refer to the same data within TreeBASE.

These cleaning tasks can be specified by high-level data cleaning scripts such as those used in [32]. For example, the VIEW operator can be specified as follows:

```
CREATE VIEW PhyloPrefixes
FROM DirtyTaxonData
WHERE d <> NULL
{SELECT prefixGen(d.TaxonName)
AS prefix INTO PhyloPrefixes }
```

Here, the prefixes are created from the table DirtyTaxonData that contains the extracted data from TreeBASE. The function prefixGen produces the prefixes from DirtyTaxonData. Other operator specifications follow similarly.

This cleaning architecture offers some advantages over traditional methods such as record-based methods. Foremost, it allows for comparison of the nomenclature



BIO-AJAX for TreeBASE –[Help]

Keyword:

Search on: Taxon

alpha A-crystallin, Nucleic Acid
Fig. 1, Amino Acid
PDHA1, Nucleic Acid
actin, Nucleic Acid
tubulin, Nucleic Acid
SSU+LSU, Nucleic Acid

12 Studies in TreeBASE:

Study #1: Barns, S.M., C.F. Delwiche, J.D. Palmer, and N. R. Pace. 1996. Perspectives on archaeal diversity, thermophily and monophyly from environmental rRNA sequences. Proc. Natl. Acad. Sci. 93: 9188–9193. [S304] Entered 9/20/98. (Authors: Susan M. Barns, Charles F. Delwiche, Norman R. Pace, Jeffrey D. Palmer).

- **Analysis**: A maximum likelihood analysis using PAUP on:
  - Matrix rDNA – [T], 1620 nucleic acid characters, 64 taxa.

  Result: one tree.
  - alv Fig. 2 (single)

Study #2: Caetano–Anollés, G. 2001. Evolved RNA secondary structure and the rooting of the universal tree of life. Journal of Molecular Evolution, 54: 333–345. [S653] Entered 8/8/01. (Authors: Gustavo Caetano–Anollés).

- **Analysis**: A parsimony analysis (named shape phylogeny) using PAUP on:
  - Matrix LSU rRNA – [T], 848 nucleic acid characters, 35 taxa.

  Result: one tree.
  - alv Fig. 6 (single)

Fig. 2 ***Obsidian Pool community.*** Surf with 1 degree of separation

Barns, S.M., C.F. Delwiche, J.D. Palmer, and N. R. Pace. 1996. Perspectives on archaeal diversity, thermophily and monophyly from environmental rRNA sequences. Proc. Natl. Acad. Sci. 93: 9188–9193.

Figure 7.3 The BIO-AJAX interface implemented for TreeBASE.

against a reliable resource (e.g. NCBI Taxonomy database) that is dynamically adjusted to the recognized state of the art within phylogenetics. Other methods for cleaning this data, such as record based comparison cleaning, would not be able to exploit such updates.

7.3.1 Implementation

The BIO-AJAX toolkit for TreeBASE is implemented using Perl, JAVA, JSP and HTML and placed over TreeBASE as middleware to preserve data integrity. Figure 7.3 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 organized 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 [8, 9, 29, 112].

The NCBI Taxonomy database [8, 9, 29, 112] 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. Moreover, it 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.

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 7.2 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 7.3. This interface has been modeled 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.

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.

For example, consider again the three trees in Figure 7.2. When the query is "Homo sapiens x" and the "Exact" button is clicked, T_1 will be returned. On the other hand, when the query is "Homo sapiens" and the "Exact" button is clicked, none of the three trees will be returned. However, if we click on the "Related" button when submitting the query "Homo sapiens", all the three trees will be returned.

7.3.2 Experiments and Results

To see how the proposed technique helps to gather a more complete set of trees related to a species, we conducted the following experiments. We extracted all of the taxon names from TreeBASE and used these taxon names as queries against NCBI TaxBrowser. This would reflect how well the TreeBASE nomenclature was compared with standard nomenclature. Moreover, the NCBI Taxonomy database also links to the TreeBASE database to provide users with more data about a given taxon. This experiment would reflect how well the taxa in the TreeBASE archive were recognized by the NCBI Taxonomy database.

For each taxon that is in a study or tree housed in the TreeBASE archive, it was used as a query for the NCBI TaxBrowser. If any data page was returned to the user, it was considered a match. This data page could be one of two types of page. First, it could be a terminal page for a specific taxon's phylogenetic information. Second, it could be a hierarchical listing of possible matches. Of the 26,737 taxa obtained from TreeBASE, four experiments were performed with the different types of search (i.e. Complete Name, Wild Card, Token Set and Phonetic Name) available on TaxBrowser.

In addition, the prefix lists were tested. 290,118 prefixes were generated for the 26,737 taxa. If any one of the prefixes for a given taxon resulted in a match within the NCBI Taxonomy database, the taxon was considered as matched. The results are shown in Table 7.1. Thus, for example, in the table, for the Complete Name search option, 13,076 taxa out of the 26,737 taxa, which were about 49% of the taxa obtained from TreeBASE, resulted in a match within the NCBI Taxonomy database. On the other hand, for the same search option and with the prefix generation technique, 23,493 taxa out of the 26,737 taxa, which were about 88% of the taxa obtained from TreeBASE, were considered as matched.

Table 7.1 Results of Taxa Searches in NCBI TaxBrowser with TreeBASE Nomenclature and Prefixes

	Search option	TreeBASE *taxa*	*With prefixes*
(i)	Complete Name	13,076 (49%)	23,493 (88%)
(ii)	Wild Card	13,079 (49%)	23,493 (88%)
(iii)	Token Set	13,172 (49%)	23,801 (89%)
(iv)	Phonetic Name	14,025 (52%)	24,633 (92%)

The results in Table 7.1 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 minimize the nomenclature problems, only approximately 50% of the taxa in TreeBASE are recognizable 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.

7.4 Conclusion and Future Work

This chapter presents BIO-AJAX, a cleaning framework for biological data. The BIO-AJAX toolkit has been implemented and placed, as middleware, over the phylogenetic information system TreeBASE. A prefix generation technique for instantiating the operators in BIO-AJAX has been developed, which helps to clean the nomenclature in TreeBASE. In contrast to existing work for cleaning relational records or genome sequences (see, for example, [31, 62]), the techniques presented here are focused on evolutionary trees and hence are different from those existing data cleaning methods.

Future work concerning this framework includes creating more data analysis tools such as viewing the statistical distributions of taxa or co-occurrence of taxa [93]. Another direction is to extend BIO-AJAX to other data repositories. This involves the implementation of an API (application programming interface) for wrapping the taxonomy source such that other taxonomy (say, enzyme classification and the Gene Ontology) can also be used so long as they adhere to the required interface. There may exist inconsistencies among the different taxonomy sources and how to resolve them remains to be a challenge research problem, both in phylogenetics and in data engineering.

Other work that can be based from BIO-AJAX for TreeBASE is protein data cleaning from the Protein Information Resource (PIR). PIR is an integrated public resource of protein informatics to support genomic and proteomic research and scientific discovery [116]. Through the development of PIR databases, a number of problems have been observed that could benefit from the BIO-AJAX data cleaning architecture. For example, in [115], two types of cleaning issues are identified: protein classification and functional annotation.

Protein classification is a complex area with a number of problems. First, there is no standard way to compare proteins. Therefore, different methods turn out different results. Another problem is that many of the protein classification issues have arisen from the data integration phase of eliminating redundancy from the repository. For example, it has been noted that there are numerous errors found in genome annotation [15, 27]. Moreover, with current genome annotation standards, many proteins are defined only to have one function. This can limit the ability to classify the proteins properly since most proteins are multifunctional. This leads to the concerns facing the functional annotation. A possible method for addressing this problem could be adopting feature extraction for proteins [103, 104] and using extracted features to categorize the proteins.

Functional annotation itself is used in many aspects of protein data repository management. For example, it can help to classify unknown protein sequences. However, current functional annotation has many problems. Besides the aforementioned problem, a major issue of functional annotation is concerned with protein name ontology [45]. Such an ontology is important, as the protein name is the form in which a protein object is referred to and communicated in the scientific literature and biological databases. As with the nomenclature problem in TreeBASE, there is also a long-standing problem of nomenclature for proteins. Scientists may name a newly discovered or characterized protein based on its function, sequence features, cellular location, molecular weight, or other properties, as well as their combinations or abbreviations. A protein name ontology provides a basis for consistent database annotation where functional conservation is curated with a common language.

The BIO-AJAX framework offers possible improvements and solutions to these basic problems. The framework allows for the extension of multiple algorithms for classifying proteins as well as creates environments for these algorithms to interact with each other. It also affords the opportunity to implement and extend new algorithms and methods. This permits for BIO-AJAX to examine how well the algorithms perform classification and interact with the older methods.

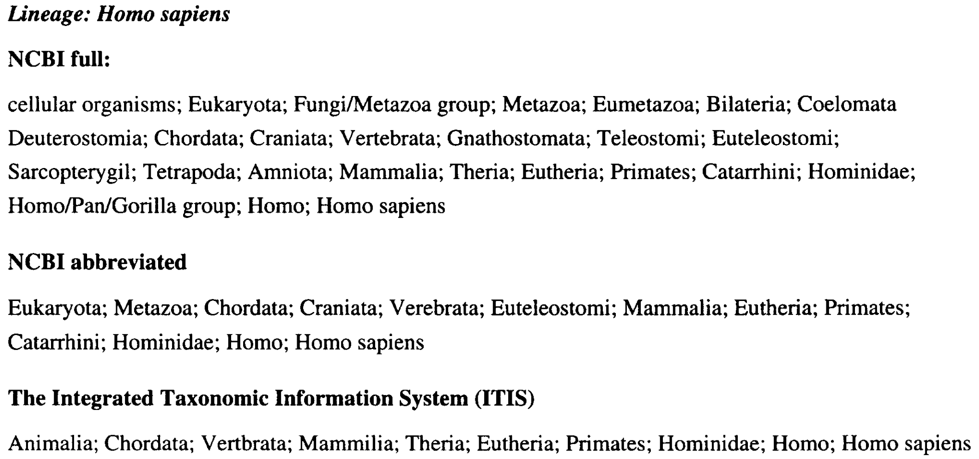
CHAPTER 8

THE BIO-AJAX TOOLKIT FOR LINEAGE PATHS

8.1 Introduction

Besides nomenclature, another key component of phylogenetic data is lineage paths. A lineage path is the path to a given point on the Tree of Life to a specific taxon. Sometimes this taxon can be a species, which tends to be a terminal node on the Tree of Life or it can be an intermediary node within the tree. Most lineage paths concern the path from the root node of the Tree of Life to a specific taxon. Lineage paths pose many different problems for phylogenetic researchers. For example, in phylogenetic nomenclature, the lineage or ranking of an evolutionary unit or taxon is not standard. NCBI Taxonomy Database uses 28 distinct ranks for classification while the International Code of Botanical Nomenclature uses 25 rankings [78]. Figure 8.1 demonstrates these differences between the NCBI Taxonomy Database and the Integrated Taxonomic Information Server [47]. Lineage paths and lineage path comparison offers phylogenetic researches many interesting functionalities for their research. Through their comparison, researchers can analyze differences in phylogenetic reconstruction methods, understand differences in rankings among phylogenetic databases as well as perform some advanced queries upon the phylogenetic studies. Some possible queries of interest associated with lineage paths include:

- Compare the lineage paths for taxon X from database D1 and database D2
- Given taxon X, find all lineage paths containing taxon X
- Given taxon X, find all taxa which are descendents of taxon X
- Given taxon Y, find all taxa which are ancestors of taxon X



Lineage: Homo sapiens

NCBI full:

cellular organisms; Eukaryota; Fungi/Metazoa group; Metazoa; Eumetazoa; Bilateria; Coelomata Deuterostomia; Chordata; Craniata; Vertebrata; Gnathostomata; Teleostomi; Euteleostomi; Sarcopterygil; Tetrapoda; Amniota; Mammalia; Theria; Eutheria; Primates; Catarrhini; Hominidae; Homo/Pan/Gorilla group; Homo; Homo sapiens

NCBI abbreviated

Eukaryota; Metazoa; Chordata; Craniata; Verebrata; Euteleostomi; Mammalia; Eutheria; Primates; Catarrhini; Hominidae; Homo; Homo sapiens

The Integrated Taxonomic Information System (ITIS)

Animalia; Chordata; Vertbrata; Mammilia; Theria; Eutheria; Primates; Hominidae; Homo; Homo sapiens

Figure 8.1 Lineage paths for Homo sapiens from the NCBI Taxonomy Database and ITIS.

Foremost, lineage path querying offers the ability to compare ranking systems among the supported databases. For example, as mentioned previously, there are differences in ranking systems from database to database. By offering lineage path querying, a user can compare side by side the differences between the paths among the different databases. These ranking are important since the scientific name of the species is dependent upon where it falls within a given ranking system. If a species is classified differently from database to database, it affects the type of data that can be retrieved about a particular taxon. A user may have an understanding of one ranking system without understanding another. By allowing for comparison, we provide the user to evaluate each databases ranking system, seeing where the data he or she is interested in may be located [74, 78].

Extending these path comparisons to phylogenetic studies, it can also help scrutinize the differences between two trees which have similar taxa but use different reconstruction methods. By breaking the trees down to their paths, we can monitor the differences in the classification of a specific taxon through various reconstruction methods. This can be useful in analyzing the effectiveness of a reconstruction method as well as determining a species proper ranking.

Finally, lineage path queries can also help with advanced querying. One extremely powerful query that no phylogenetic database support effectively is given a node N of the phylogenetic tree, find all ancestors or descendents from N. To perform this type of search, most databases require that the user navigate through some hierarchical browsing method of analyzing the tree. To find a specific path of the tree, the user must be familiar with the tree. For example, if a novice user tries to find "Homo sapiens from the root of the tree, most novices would not understand enough about phylogenetics to know that the first classification "Homo sapiens falls under is "Eukaryota . Therefore, a user can enter a portion of the path, for example "Homo sapiens, and get the results he or she wants. Also, if the user is unclear about what type of organisms would fall under "Eukaryota, he or she can enter "Eukaryota as a partial path query, receiving back all paths that contain "Eukaryota.

8.2 The BIO-AJAX Toolkit for Lineage Paths

The BIO-AJAX toolkit facilitates improving the state of lineage path querying through integrating lineage path resources. By applying the cleaning architecture, various lineage path resources can be integrated together to offer the user the ability to query and compare these lineage paths.

Specifically, the operators of BIO-AJAX are instantiated to clean and integrate the lineage path in the following manner (CLASSIFY is not relevant here and is omitted):

- MAP: produces, from the list of taxa and paths, a list of suffixes for paths.
- VIEW: generates a list of all taxa associated with their lineage paths from the lineage path repositories
- MATCH: compares the suffixes to find identical or similar paths.

- MERGE: Searches through the lineage paths. If two lineage paths are identical or similar, merges the two paths.

- CLUSTER: Groups all suffixes generated lineage paths based on the first taxon in the suffix.

The BIO-AJAX framework for Lineage Paths has been implemented using the NCBI Taxonomy Database and the Integrated Taxonomic Information Server (ITIS). Both tools allow querying upon their taxon set and return lineage paths as a part of the answer to the query. In the current implementation of BIO-AJAX for lineage paths, the following queries are addressed:

- Compare the lineage paths for taxon X from database D1 and database D2

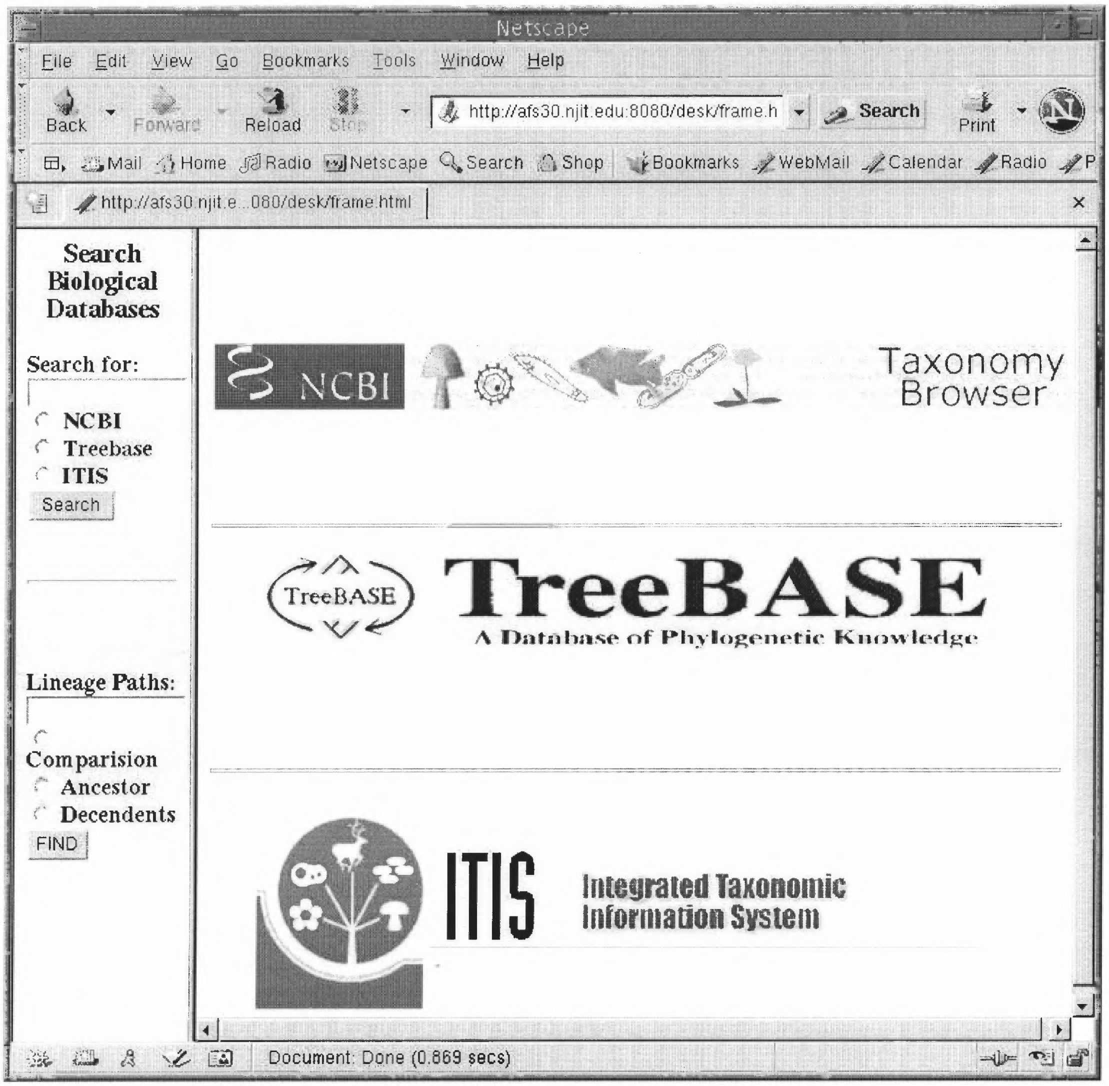
- Given taxon X, find all taxa which are descendents of taxon X

- Given taxon Y, find all taxa which are ancestors of taxon X

In future versions of this tool, the option of finding any lineage path that contains taxon X will be provided. Moreover, other databases repositories lineage paths will also be incorporated into the integrated framework.

8.3 Implementation

The current implementation for BIO-AJAX for Lineage Paths uses the data warehousing technique for integrating the data and is accessible through a World Wide Web interface. Figure 8.2 displays the interface for this tool. The lineage paths are extracted both from NCBI Taxonomy Database and ITIS and stored locally. In previous version of the tool, 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


Netscape
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
http://afs30.njit.edu:8080/desk/frame.h
Search
Print
Mail Home Radio Netscape Search Shop Bookmarks WebMail Calendar Radio P
http://afs30.njit.e...080/desk/frame.html
Search Biological Databases
Search for:
NCBI
Treebase
ITIS
Search
Lineage Paths:
Comparision
Ancestor
Decendents
FIND
NCBI
Taxonomy Browser
TreeBASE
TreeBASE
A Database of Phylogenetic Knowledge
ITIS
Integrated Taxonomic Information System

Document: Done (0.869 secs)

Figure 8.2 Interface for BIO-AJAX for Lineage Paths.

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.

8.3.1 Lineage Path Extraction

For both data repositories, lineage paths can be extracted from the flat files each provides as a free download. While both databases vary in format, both store the lineage path information in similar ways. Each database has signified one taxon as a "root for the tree of life its data model represents. Each taxon has various types of information stored about it, including the taxon id number of its immediate ancestor. Therefore, to obtain any taxons lineage path, a user would traverse the flat file recursively, until he or she obtains the root taxon.

Initially, the flat file from the repository is downloaded to local storage. Next, the scientific nomenclature is associated with its taxon identification number. To facilitate traversal of the paths, all taxon are also indexed through their taxon identification number in an array, with the taxon identification number being offset to the array index number. This array uses the taxons identification number as its index and then stores within the array the taxons parents taxon identification number. This allows for quick retrieval as well as facilitates recursive calls to obtain the next parent. Since these files contain approximately 300,000 taxa, this array indexing is separated into files of 50,000 taxa. At the initiation of the program, the program checks how many individual taxa are within the file and scales the number of arrays accordingly. When traversing through the lineage path, the appropriate array in brought into main memory to retrieve the appropriate parent taxon identification number. Once the parents identification number is retrieved, it is stored in a file associated with the original taxon. Next, the parents identification number becomes the index searched for and its parent is retrieved. If the parents identification number is within the same array currently loaded within memory, then that array is searched. Otherwise, the appropriate array data is brought into memory and it is search accordingly. Once the path has been traversed to the root, the next taxon from the repositorys original

ITIS Lineage Path for Homo sapiens

Animalia; Chordata; Vertbrata; Mammilia; Theria; Eutheria; Primates; Hominidae; Homo; Homo sapiens

Path Representation:

```
0         1         2         3         4       5        6          7          8      9
Animalia#Chordata#Vertbrata#Mammilia#Theria#Eutheria#Primates#Hominidae#Homo#Homo_sapiens
```

Sorted Suffixes

0 Animalia#Chordata#Vertbrata#Mammilia#Theria#Eutheria#Primates#Hominidae#Homo#Homo_sapiens

1 Chordata#Vertbrata#Mammilia#Theria#Eutheria#Primates#Hominidae#Homo#Homo_sapiens

7 Hominidae#Homo#Homo_sapiens

8 Homo#Homo_sapiens

9 Homo_sapiens

5 Eutheria#Primates#Hominidae#Homo#Homo_sapiens

3 Mammilia#Theria#Eutheria#Primates#Hominidae#Homo#Homo_sapiens

6 Primates#Hominidae#Homo#Homo_sapiens

4 Theria#Eutheria#Primates#Hominidae#Homo#Homo_sapiens

2 Vertbrata#Mammilia#Theria#Eutheria#Primates#Hominidae#Homo#Homo_sapiens

Figure 8.3 Suffix representation for the lineage path for Homo sapiens.

flat file is obtained and the process begins again. This process is performed for every taxon within the repositorys flat file.

Once the paths have been outputted to a flat file, the paths are in numeric format, with each taxon represented by its taxon identification number. Each path is reversed so that the root is the first taxon read and the taxon for which this is the lineage path is the last taxon read. Next, the nomenclature is substituted into the path for the taxon identification numbers. These paths are they rewritten back to flat file format.

To store this information, each lineage path is stored in a hash table indexed by the taxon name with an indication of which database the taxon was obtained from.

8.3.2 Lineage Path Indexing and Retrieval

Querying for the purpose of comparison, for finding all ancestors and for finding all descendents poses different problems that must be addressed so that the query is executed properly. Concerning querying for comparison and querying to find all ancestors, the data returned is very similar for both queries. However, for finding all descendents, the path must be manipulated in a different way to obtain these answers both efficiently and effectively.

The comparison query and the ancestors query can be effectively queried using the hash index from the extraction phase. Since the paths from the extraction phase are indexed according to the taxon name, the paths can easily be retrieved. Moreover, due to the nature of both databases, each taxon has a unique lineage path associated with it. The NCBI Taxonomy Database does offer the ability to abbreviate the path, but in the integration with this database, only the full path is considered.

For the comparison query, the paths are displayed vertically with the leaf taxon at the end. This allows for the user to compare both paths side by side. Figure 8.4 demonstrates how these paths are displayed. For the ancestor query, the same paths for the comparison query are obtained. However, in this case, the query taxon is not shown but only its ancestor taxa.

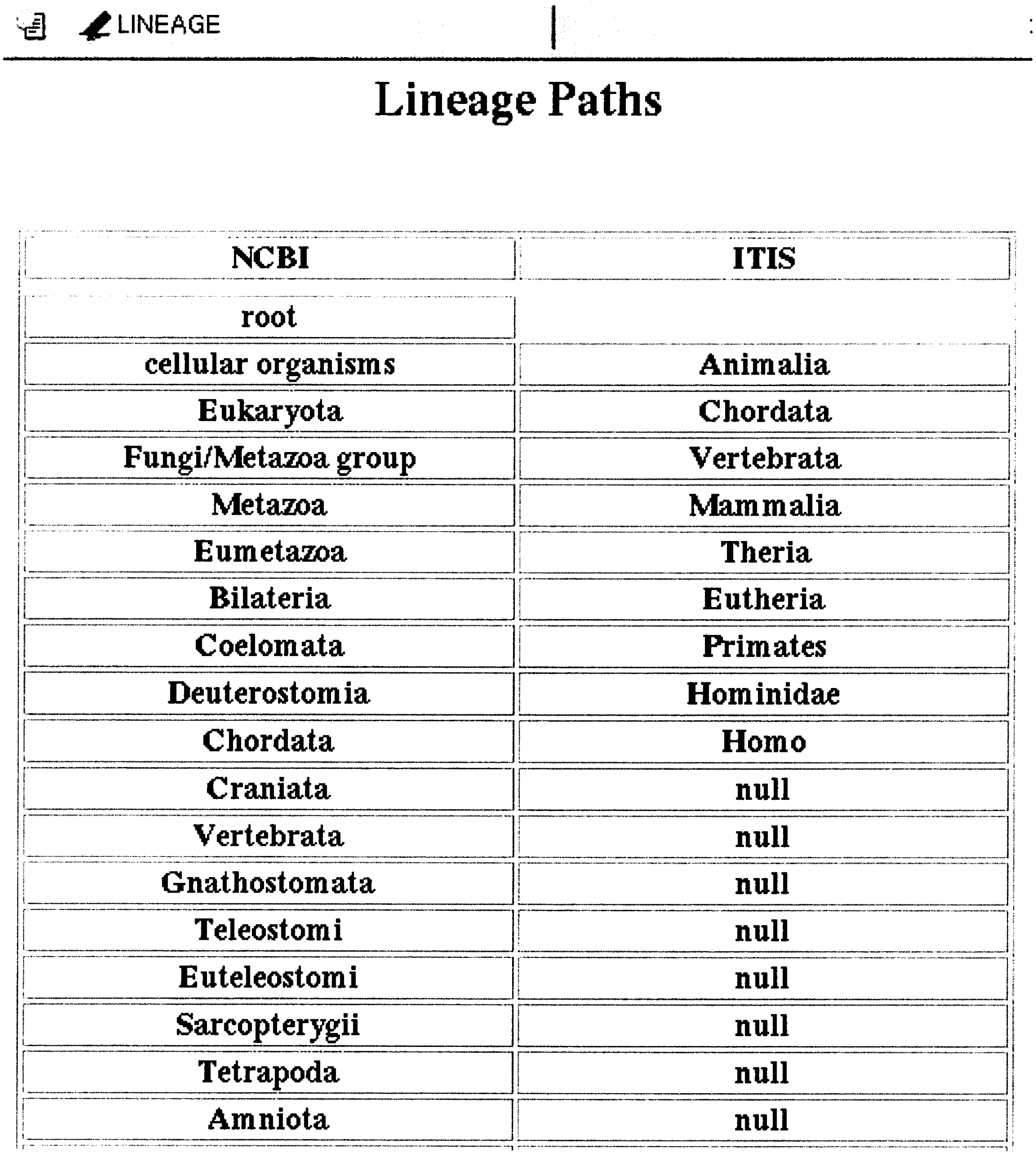
For the third query, where the interest is finding all descendents of a given node, the hash structure is unable to provide this. Therefore, suffix arrays [66] have been used to index the data. A suffix array is a data structure that, for a given string, stores all pointers to suffixes for that string lexographically in an array. Then, when querying the array, the user can search the array using a log N search algorithm such as binary search to find the data he or she is hunting for. In this application of suffix arrays, all suffixes for a given lineage path are generated. A suffix for a lineage path is generated by taking the all suffixes, starting at the first letter of a taxon within the lineage path. For example, Figure 8.3 demonstrates how the suffixes are generated with their pointers.

After the suffixes are generated, they are clustered together with all suffixes that begin with the same taxon. Since the first taxon in the suffixes represents the ancestor of every other taxon within the path, all of the descendents from the leading taxon can be obtained from these suffixes. The suffixes are then sorted and stored within the suffix array associated with that taxon. These files are then indexed according to that leading taxon. When a descendent query is executed, the tool finds the array associated with that taxon. The first taxon is removed from each path, thereby giving the descendents for that taxon.

8.3.3 Data Cleaning and the Suffix Array Implementation

In the current method for finding the descendents, a number of taxon can possibly be returned numerous times, depending upon how close the query taxon is to the root of the tree. Therefore, data cleaning methods can be applied to these paths to eliminate redundancy.

One possible method for eliminating redundancy is to apply the sorted neighborhood method to the array to detect similarities. Since the paths are highly structured and common errors such as spelling errors are rare, this becomes an ideal application for using the sorted neighborhood method. In a simple implementation of sorted neighborhood method on lineage paths, at the very least common paths between taxa on the same level of the tree can be found by comparing the lineage paths up to but not including the final taxon in the path. Identical paths can be merged so that redundancy is eliminated. For more advanced applications, there can be a number of iterations of this comparison, where the first iteration starts by comparing for one common taxon and builds to more complex paths.



Lineage Paths

NCBI	ITIS
root	
cellular organisms	Animalia
Eukaryota	Chordata
Fungi/Metazoa group	Vertebrata
Metazoa	Mammalia
Eumetazoa	Theria
Bilateria	Eutheria
Coelomata	Primates
Deuterostomia	Hominidae
Chordata	Homo
Craniata	null
Vertebrata	null
Gnathostomata	null
Teleostomi	null
Euteleostomi	null
Sarcopterygii	null
Tetrapoda	null
Amniota	null

Figure 8.4 Path comparison output.

8.3.4 Lineage Path Retrieval

To execute these three queries, a user can interact with the indices through a web based search mechanism. On this web page a user enters a taxon name within a text box. He or she next selects from a radio button list beneath the text box what type of query he or she wants executed. From this, the appropriate the appropriate query is executed.

8.4 Conclusion and Future Work

Lineage paths offer an interesting and rich method for phylogenetic researchers to explore various interpretations of the Tree of Life. By creating a comparative environment for phylogenetic researchers, problems concerning the correctness of the Tree of Life can be addressed.

Future work concerning this research includes integrating more repositories into the tool as well as improving the user interface. Moreover, this work can be further applied to consensus tree and supertree generation problems. Also, work can be done to further the visualization aspects of this tool concerning the comparisons.

CHAPTER 9

IMPACTS, FUTURE WORK AND CONCLUSION

9.1 Impact on Biological Data Mining and Information Retrieval

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 standardized format for whatever processing the users need. Since the data, after cleaning, has been organized 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 organized from the cleaning operations. Through the mapping, matching and merging operations, data will be transformed into the schema that the knowledge discovery tools applied to the data can analyze 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 minimized 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 standardized 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 organized properly, data mining algorithms can see these errors as interesting aberrations within the data rather than errors [23, 24]. 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 [23, 24].

9.2 Future Work

This dissertation offers starting points for both the BIO-AJAX framework as well as the two tools extended from it. These extensions can explore both the possibilities of applying BIO-AJAX to other data sets as well as creating more powerful phylogenetic tools.

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 TreeBASE. This thesis 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 standardization 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 detection, 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 dissertation 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.

9.3 Conclusion

This dissertation presents the concepts issues concerning data cleaning and biological data. It presents an overview of data cleaning and the currently popular methods used to approach this problem. It then goes on to discuss the data cleaning issues related to biological databases. It next describes one possible framework for data cleaning and introduces the concepts behind the proposed biological data cleaning system BIO-AJAX. BIO-AJAX is intended to be a flexible framework for performing data cleaning on biological data and biological databases. The work was originally inspired by the work of Galhardas et al. in "Declarative Data Cleaning: Language, Model and Algorithms " [32, 33] and the AJAX data cleaning system. Future work on this system will be to expand upon the operators, precisely defining their various functions for various biological data sets. These definitions will them be extended so that data cleaning can be performed effectively on various biological data sets such as protein data, microarray data and continued instantiation upon phylogenetic data.

REFERENCES

[1] Achard, F., Vaysseix, G., and Barillot, E.. XML, bioinformatics and data integration, *Bioinformatics.*17(2001)115-125.

[2] Agrawal, R., Imielinski, T., and Swami, A., Mining association rules between sets of items in large databases, in *Proc. 1993 ACM-SIGMOD Int. Conf. on Management of Data*, Buneman, P. and Jajodia, S., Eds., ACM Press, Washington, D.C., 1993, 207.

[3] Agrawal, R. and Srikant, R., Fast algorithm for mining association rules, in *Research Report RJ 9839*, IBM Almaden Research Center, San Jose, CA, June 1994.

[4] Agrawal, R. and Srikant, R., Fast algorithm for mining association rules, in *Proc. of the 20th Int. Conf. on Very Large Databases*, Bocca, J.B., Jarke, M. and Zaniolo, C., Eds., Morgan Kaufman, Santiago, Chile, 1994, 24.

[5] Agrawal, R., Gehrke, J., Gunopulos, D., and Raghavan, P., Automatic subspace clustering of high dimensional data for data mining, in *Proc. of the ACM-SIGMOD Conf. on Management of Data*, Tiwary, A. and Franklin, M., Eds., ACM Press, Seattle, Washington, 1998, 94.

[6] Arasu, A., Cho, J., Garcia-Molina, H., Paepcke, A. and Raghavan, S., Searching the Web, *ACM Transactions on Internet Technology*, Kim, W., Ed., ACM Press, 2001, 2.

[7] Baeza-Yates, R. and Ribeiro-Neto, B., *Modern Information Retrieval*, ACM Press, Addison-Wesley, New York, New York, 1999.

[8] Benson, D. A., M. S. Boguski, D. J. Lipman, Ostell, J. and B. F. Ouellette. GenBank, *Nucleic Acids Research.* 26(1998): 1-7.

[9] Benson, D.A., Karsch-Mizrachi, I., Lipman, D.J., Ostell, J., Rapp, B.A. and Wheeler, D.L. GenBank, *Nucleic Acids Research*, 28 (1999) : 15-18.

[10] Berners-Lee,T, Hendler, J. and Lassila, O. A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities, *Scientifc American*, May 2001.

[11] Berman, H.M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T.N., Weissig, H. ,Shindyalov, I.N. and Bourne, P.E. The Protein Data Bank, *Nucleic Acids Research*, 28 (2000): 235-242.

[12] Bitton, D. and DeWitt, D.J. Duplicate Record Elimination in Large Data Files, *ACM Transactions on Databases* 8.2 (1983): 255-265.

[13] Blake, J.A. and Harris, M. 2003. The Gene Ontology Project: Structured vocabularies for molecular biology and their application to genome and expression analysis, in *Current Protocols in Bioinformatics*, Baxevanis, A.D., Davison, D.B., Page, R., Stormo, G. and Stein, L., Eds., Wiley and Sons, Inc., New York.

[14] Breiman, L., Friedman, J.H., Olshen, R.A. and Stone, C.J., *Classification and Regression Trees*, Wadsworth Publishing Company, Statistics/Probability Series, 1984.

[15] Brenner, S.E. Errors in Genome Annotation, *Trends in Gen.*, 15(1999)132-133.

[16] Calí, A., De Giacomo, G. and Lenzerini. M. Models for information integration: Turning local-as-view into global-as-view, in *Proc. of Int. Workshop on Foundations of Models for Information Integration*, 2001.

[17] Caruso, F.,Cochinwala, M., Ganapathy, U., Lalk, G. and Missier, P. Telcordia's Database Reconciliation and Data Quality Analysis Tool, in *Proc. of 26th International Conference on Very Large Data Bases*, September 10-14, 2000, Cairo, Egypt: 615-618.

[18] Chang, G., Healey, M.J., McHugh, J.A.M., and Wang, J.T.L., *Mining the World Wide Web: An Information Search Approach*, Kluwer Academic Publishers, Boston, 2001.

[19] Cleveland, W., *Visualizing Data*, Hobart Press, Summit, New Jersey, 1993.

[20] Cochinwala, M., Kurien, V., Lalk, G. and Shasha, D. Efficient data reconciliation, *Information Sciences*, 137:1-4 (Sept. 2001):1-15.

[21] Cole, J.R., Chai, B., Marsh, T.L., Farris, R.J., Wang, Q., Kulam, S.A., Chandra, S., McGarrell, D.M., Schmidt, T.M., Garrity, G.M. and Tiedje, J.M. The Ribosomal Database Project (RDP-II): previewing a new autoaligner that allows regular updates and the new prokaryotic taxonomy, *Nucleic Acids Research*, 31:1(2003):442-3.

[22] Croft, W.B., Turtle, H.R., and Lewis, D.D. The use of phrases and Structured queries in information retrieval, in *Proc. 14th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, Bookstein, A. et al., Eds., ACM Press, Chicago, Illinois, 1991, 32.

[23] Dasu, T. and Johnson, T. Problems, Solutions & Research in Data Quality, in *Proc. Of the SIAM International Conference on Data Mining*, Arlington, Virginia, April 11-13, 2002.

[24] Dasu, T. and Johnson, T. *Exploratory Data Mining and Data Cleaning.* John Wiley & Sons, 2003.

[25] Davidson, B., Overton, C. and Buneman, P. Challenges in Integrating Biological Data Sources, *Journal of Computational Biology*,2(1995):557-572.

[26] Davidson, S.B., Overton Tannen, V. and Wong, L. BioKleisli: A Digital Library for Biomedical Researchers, *Int. J. on Digital Libraries*,1(1997):36-53.

[27] Devos, D. and Valencia, A. Intrinsic Errors in Genome Annotation, *Trends in Gen.*, 17(2001):429-431.

[28] Ester, M., Kriegel, H.P., Sander, J. and Xu, X., A density-based algorithm for discovering clusters in large spatial databases, in *Proc. 1996 Int. Conf. Knowledge Discovery and Data Mining(KDD'96)*, Simoudis, E., Han, J., Fayyad, U.M., Eds., ACM Press, Portland, Oregon, 1996, 226.

[29] Federhen, S., Harrison, I., Hotton, C., Leipe, D., Soussov, V., Sternberg, R., and Turner, S. NCBI Taxonomy Homepage, `http://www.ncbi.nlm.nih.gov/Taxonomy/taxonomyhome.html/`, 30 April 2004.

[30] Frakes, W.B. and Baeza-Yates, R., Eds., *Information Retrieval: Data Structures and Algorithms*, Prentice–Hall, Englewood Cliffs, NJ, 1992.

[31] Gajer, P., Schatz, M., and Salzberg, S.L. Automated correction of genome sequence errors, *Nucleic Acids Research*, 32(2004): 562-569.

[32] Galahardas, H., Florescu, D., Shasha, D., Simon, E. and Saita, C.A. Declarative Data Cleaning: Language, Model and Algorithms, in *Proc. of 27th International Conference on Very Large Data Bases*, September 11-14, 2001, Roma, Italy : 371-380.

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

[34] The Global Biodiversity Information Facility, `http://www.gbif.org/`, 30 April 2004.

[35] Greer, D.S., Bourne, P.E. and Berman, H.M. The Protein Data Bank: unifying the archive, *Nucleic Acids Research.* 30(2002): 245-248.

[36] Halevy, A.Y. Answering queries using views: A survey, *Very Large Database Journal*, 10:4(2001): 270-294.

[37] Halevy, A.Y. Data Integration: A Status Report. *BTW 2003*: 24-29, Gerhard Weikum, Harald Schning, Erhard Rahm (Eds.): BTW 2003, Datenbanksysteme fr Business, Technologie und Web, Tagungsband der 10. BTW-Konferenz, 26.-28. February 2003, Leipzig.

[38] Halevy, A.Y., Ives, A.G., Mork, P., Tatarinov, I.: Piazza: data management infrastructure for semantic web applications, in *Proc. of the Twelfth International World Wide Web Conference*, WWW2003, Budapest, Hungary, 20-24 May 2003. ACM, 2003, 556-567.

[39] Han, J. and Kamber, M., *Data Mining: Concepts and Techniques*, Morgan Kaufmann, San Francisco, California, 2000.

[40] Harlin, M. Taxon names as paradigms: the structure of nomenclatural Revolutions, *Cladistics.* 19(2003): 138-143.

[41] Herbert, K.G., Gehani, N.H., Wang, J.T.L., Piel, W.H., Wu, C.H. BIO-AJAX: An Extensible Framework for Biological Data Cleaning. *ACM SIGMOD Record Special Issue on Data Management in Life Sciences.* To Appear.

[42] Herbert, K.G., Westbrook, J., Wang, J.T.L. Data Integration in Biological Database, in *Proc. of the Atlantic Symposium on Computational Biology and Genome Information Systems & Technology*, Durham, North Carolina, September, 2003.

[43] Hernandez, M.A. and Stolfo, S.J. The merge/purge problem for large databases, in *Proc. of the 1995 ACM SIGMOD International Conference on Management of data*, May 22-25, 1995, San Jose, California, United States: 127-138.

[44] Hernandez, M.A. and Stolfo, S.J. Real-world Data is Dirty: Data Cleansing and the Merge/Purge Problem, *Data Mining and Knowledge Discovery*, 2(1998):9-37.

[45] Hirschman, L., Park, J.C., Tsujii, J., Wong, L., and Wu, C.H. Accomplishments and Challenges in Literature Data Mining for Biology. *Bioinformatics*,18:12(2002):1553-1561.

[46] Huang, H., Barker, W.C., Chen, Y., and Wu, C.H.. iProClass: an integrated database of protein family, function and structure information, *Nucl. Acids. Res.* 31(2003):390-392.

[47] The ITIS Organization, The Integrated Taxonomic Information System (ITIS) `http://www.itis.usda.gov/`, 2 May 2004.

[48] Ives, Z., Levy, A.Y., Madhavan, J., Pottinger, R., Saroiu, S., Tatarinov, I., Betzler, S, Chen, Q., Jaslikowska, E., Su, J.,Yeung, W.T.T. Self-Organizing Data Sharing Communities with SAGRES. In *Proc. of the 2000 ACM SIGMOD International Conference on Management of Data*, May 16-18, 2000, Dallas, Texas, USA. ACM 2000, 582.

[49] Johnson, R.A. and Wichern, D.A., *Applied Multivariate Statistical Analysis*, Prentice Hall, Upper Saddle River, New Jersey, 1992.

[50] Kartoo website, `http://www.kartoo.com`, 30 April 2004.

[51] Kaufman, L. and Rousseeuw, P.J., *Finding Groups in Data: An Introduction to Cluster Analysis*, John Willey & Sons, New York, New York, 1990.

[52] Kobayashi, M. and Takeda, K., Information retrieval on the web, *ACM Computing Surveys*, Wegner, P. and Israel, M., Eds., ACM Press, 144, 2000.

[53] Korfhage, R., *Information Storage and Retreival*, John Wiley & Son, Inc., 1997.

[54] Krause, A., Haas, S.A., Coward, E. and Vingron, M., SYSTERS, GeneNest, SpliceNest: exploring sequence space from genome to protein, *Nuc. Acids Res.*, Oxford University Press, 299, 2002.

[55] Lacroix, Z.. Biological data integration: wrapping data and tools,*IEEE Trans. on Information Technology in Biomedicine*,6(2002),123-128.

[56] Lacroix, A. The Biological Data Integration System, in *Proceedings of the fifth ACM international workshop on Web information and data management*, New Orleans, Lousiana, 2003, 45-49.

[57] Lacroix, Z. and Critchlow, T., Eds., *Bioinformatics: Managing Scientific Data*, Morgan Kaufmann, 2003.

[58] Lay, J.A., Muneesawang, P., and Guan, L., Multimedia information retrieval, in *Proc of Canadian Conference on Electrical and Computer Engineering*, Dunne, S., Eds., IEEE Press, Toronto, Canada, 2001, 619.

[59] Lenzerini, M. Data integration: a theoretical perspective, in *Proc. of the 21st ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, Babcock et. al, Eds., ACM Press, Madison, Wisconsin, 2002: 233-246.

[60] Leroy, G. and Chen, H., Meeting medical terminology needs: The ontology-enhanced Medical Concept Mapper, *IEEE Transactions on Information Technology in Biomedicine*, 5:4(Dec 2001): 261-270.

[61] Leroy, G., Chen, H., Genescene: Biomedical Text And Data Mining, *JCDL 2003*: 116-118.

[62] Low, W.L., Lee, M.L., and Ling, T.W. A knowledge-based approach for duplicate elimination in data cleaning, *Information Systems*, 26:8(Dec. 2001): 585-606.

[63] Ludäscher, B., Gupta, A., and Martone, M.E. A Model Based Mediator System for Scientific Data Management, Lacroix, Z. and Critchlow, T., Eds., *Bioinformatics: Managing Scientific Data*, Morgan Kaufmann Publishers, 2003,335-370.

[64] MacQueen, J., Some methods for classification and analysis of multivariate observation, in *Proc. 5th Berkeley Symp. Math. Statist, Prob.*, Le Cam, L.M and Neyman, J., Eds., University of California Press, Berkeley, 1967, 281.

[65] Maddison, D.R. Tree of Life, `http://tolweb.org/tree/phylogeny.html.`, 1 May 2004.

[66] Manber, U, Myers, E.W., Suffix Arrays: A New Method for On-Line String Searches, *SIAM Journal of Computing* 22:5(1993): 935-948.

[67] Mannila, H., Toivonen, H., and Verkamo, A.I., Efficient algorithms for discovering association rules, in *Proc. of the AAAI Workshop on Knowledge Discovery in Databases*, Fayyad, U.M., Uthurusamy, R., Eds., AAAI Press, Seattle, Washington, 1994, 181.

[68] Meghini, C., Sebastiani, F., and Straccia, U., A model of multimedia information retrieval, *Journal of the ACM* , ACM Press, 48 909, 2001.

[69] Mobasher, B., Dai, H., Luo, T., Nakagawa, M.:, Web data mining: Effective personalization based on association rule discovery from web usage data, in *Proc of the 3rd Int. Work. on Web information and data management*, ACM Press, Atlanta, Georgia, 2001, 243.

[70] Müller, H., Naumann, F. Data Quality in Genome Databases, in *Proc. of the Eighth International Conference on Information Quality (IQ 2003)*, November 7-9. 2003, 269-284.

[71] Murthy, S.K., Automatic construction of decision trees from data: A multi-disciplinary survey, *Data Mining and Knowledge Discovery*, Kluwer Academic Publishing, 2, 345, 1998.

[72] Nack, F. and Lindsay, A., Everything you wanted to know about MPEG-7: Part 1, *IEEE Multimedia*, IEEE Press, July-September 1999, 65.

[73] Nack, F. and Lindsay A., Everything you wanted to know about MPEG-7: Part 2, *IEEE Multimedia*, IEEE Press, October-December 1999, 64.

[74] Nakhleh, L., Miranker, D.P., Barbancon, F., Piel, W.H., Donoghue, M. Requirements of Phylogenetic Databases, in *Proc. Of the 3rd IEEE International Symposium on BioInformatics and BioEngineering*, 10-12 March 2003, Bethesda, MD: 141-148.

[75] Nie, Z. andKambhampati, S. A Frequecny-based Approach for Mining Coverage Statistics in Data Integration, in *Proc. of the Internation Conference on Data Engineering*, 30 March 2004 2 April 2004, Boston, Massachusetts.

[76] Nie, Z., Kambhampati, S. and Hernandez, T. BibFinder/StatMiner: Effectively Mining and Using Coverage and Overlap Statistics in Data Integration, in *Proc of the 29th International Conference on Very Large Data Bases*, September 9-12, 2003, Berlin, Germany : 1097-1100.

[77] Page, L., Brin, S., Motwani, R. and Winograd, T., The pagerank citation ranking: Bringing order to the Web, *Tech. Rep. Computer Systems Laboratory*, Stanford University, Stanford, CA., 1998.

[78] Page, R.D.M.: Phyloinformatics: Towards a Phylogenetic Database, in *Data Mining in Bioinformatics*, Wang, J.T.L, et al., Eds. To appear.

[79] Page, R.D.M. and Holmes, E.C. *Molecular Evolution: A phylogenetic approach.* Oxford: Blackwell Scientific, 1998.

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

[81] Piel, W.H., Sanderson, M.J., and Donoghue, M. The Small-world Dynamics of Tree Networks and Data Mining in Phyloinformatics, *Bioinformatics*, 19(9):1162-1168, 2003.

[82] Rahm, E. and Do, H.H. Data Cleaning: Problems and Current Approaches , in *Bulletin of the Technical Committee on Data Engineering, Special Issue on Data Cleaning.* 23.4 (Dec 2000): 3-13.

[83] Raman, V. and Hellerstein, J.M. Potter's Wheel: An Interactive Data Cleaning System, in *Proceedings of 27th International Conference on Very Large Data Bases*, September 11-14, 2001, Roma, Italy: 381-390.

[84] Riloff, E., Little words can make a big difference for text classification, in *Proc. 18th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, Fox, E.A., Ingwersen, P. and Fidel, R., Eds., ACM Press, Seattle, Washington, 1995, 130.

[85] Riloff, E. and Hollaar, L.A., Text Databases and Information Retrieval, *The Computer Science and Engineering Handbook*, Tucker, A.B., Ed., CRC Press, 1997, 1125-1141.

[86] Salton, G., *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*, Addison–Wesley, Reading, Massachusetts, 1989.

[87] Salton, G., Wong, A., and C.S. Yang, A vector space model for automatic indexing, in *Communications of the ACM*, Ashenhurst, R., Ed., ACM Press, 18, 613, 1975.

[88] Sanderson, M.J., Purvis, A. and Henze, C. Phylogenetic supertrees: assembling the trees of life. *Trends in Ecology & Evolution*, 13.3 (March 1998): 105-109.

[89] Shan, H., Herbert, K.G., Piel, W., Shasha, D., and Wang, J.T.L. A Structure-Based Search Engine for Phylogenetic Databases, in *Proc. Of the 14th International Conference on Scientific and Statistical Database Management*, July, 2002, Edinburgh, Scotland.

[90] Sharkey, M.J. and Leathers, J.W. Majority Does Not Rule: The Trouble with Majority-Rule Consensus Trees. *Cladistics.* 17.3 (Sept. 2001): 282-284.

[91] Shasha, D., Wang, J.T.L., and Giugno, R., Algorithmics and applications of tree and graph searching, in *Proc. of the 21st ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, Abiteboul, S., Kolaitis, P.G., Popa, L., Eds., ACM Press, Madison, Wisconsin, 2002, 39.

[92] Shasha, D., Wang, J.T.L., Shan, H., Zhang, K. AtreeGrep: Aproximate Searching in unordered trees, in *Proceedings of 14th International Conference on Scientific and Statistical Database Management*, Edinburgh, Scotland, UK, July 2002: 89-98.

[93] Shasha, D., Wang, J.T.L., and Zhang, S. Unordered Tree Mining with Applications to Phylogeny. In *Proc. of the 20th International Conference on Data Engineering*, 30 March 2004 2 April 2004, Boston, Massachusetts.

[94] Species 2000. `http://www.sp2000.org/`, 1 May 2004.

[95] Stevens, R., Goble, C., Paton, N., Bechhofer, S., Ng, G., Baker, P., and Brass, A. Complex Query Formulation Over Diverse Information Sources in TAMBIS, in *Bioinformatics: Managing Scientific Data*, Lacroix, Z. and Critchlow, T., Eds., Morgan Kaufmann, Chapter 7, pp. 189–224, 2003.

[96] Stockham, C., Wang, L.S., and Warnow, T. Statistically Based Postprocessing of Phylogenetic Analysis by Clustering. Bioinformatics Discovery Notes. 1.1 (2002): 1-9.

[97] Stoeckert, Jr., C.J., Salas, F., Brunk, B., and Overton, G.C. EpoDB: a prototype database for the analysis of genes expressed during vertebrate erythropoiesis, *Nucl. Acids. Res.*,27,200,1999.

[98] Stoesser G., Baker, W., van den Broek, A., Garcia-Pastor, M., Kanz, C., Kulikova, T., Leinonen, R., Lin, Q., Lombard, V., Lopez, R., Mancuso, R., Nardone, F., Stoehr, P., Tuli, M.A., Tzouvara, K. and Vaughan, R. The EMBL Nucleotide Sequence Database: major new developments Nucleic Acids Research. *Nucl. Acids Res.*, 311 (2003): 17-22.

[99] Tateno, Y., Fukami-Kobayashi, K., Miyazaki, S., Sugawara, H. and Gojobori, T. DNA Data Bank of Japan at work on genome sequence data. *Nuc. Acids Res.* 26.1 (1998): 16-20.

[100] Vassiliadis, P., Vagena, Z., Skiadopoulos, S., Karayannidis, N. and Sellis, T. K. ARKTOS: towards the modeling, design, control and execution of ETL processes, *Information Systems.* 26.8 (2001): 537-561.

[101] Vassiliou, Y. Foundations of Data Quality Espirit Project. `http://www.dbnet.ece.ntua.gr/~dwq/` , 30 April 2004.

[102] Vivisimo website, `http://www.vivisimo.com`, 30 April 2004.

[103] Wang, J.T.L., Ma, Q., Shasha, D., and Wu, C.H. New Techniques for Extracting Features from Protein Sequences, *IBM Systems Journal, Special Issue on Deep Computing for the Life Sciences*, 40(2):426-441, 2001.

[104] Wang, J.T.L., Marr, T.G., Shasha, D., Shapiro, B.A., Chirn, G.W., and Lee, T.Y. Complementary Classification Approaches for Protein Sequences, *Protein Engineering*, 9(5):381-386, 1996.

[105] Wang, J.T.L, Shapiro, B.A., and Shasha, D., Eds., *Pattern Discovery in Biomolecular Data: Tools, Techniques, Applications*, Oxford University Press, New York, 1999.

[106] Wang, J.T.L., Shan, H., Shasha, D. and Piel, W.H. TreeRank: A Similarity Measure for Nearest Neighbor Searching in Phylogenetic Databases, in *Proc. of the 15th International Conference on Scientific and Statistical Database Management (SSDBM 2003)*, Cambridge, Massachusetts, July 2003. 171-180.

[107] Wang, J.T.L., Zhang, K., Jeong, K. and Shasha, D. A System for Approximate Tree Matching, *IEEE Transactions on Knowledge and Data Engineering.* 6.4 (1994): 559-571.

[108] Wang, R.Y., Ziad, M., Lee , M.Y. *Data Quality.* Kluwer Academic Publishers, 2001.

[109] Wang, W. and Yang, J. and Muntz, R., STING: A statistical information grid approach to spatial data mining, in *Proc. 1997 Int. Conf. Very Large Databases (VLDB'97)*, Jarke, M. et al., Eds., Morgan Kaufman, Athens, Greece, 1997, 186.

[110] Weiss, S.M. and Kulikowski, C.A., *Computer Systems that Learn: Classification and Prediction Methods for Statistics, Neural Nets, Machine Learning, and Experts Systems*, Morgan Kaufmann, San Francisco, California, 1991.

[111] Wen, J.R, Nie, J.Y., and Zhang, H.J., Clustering user queries of a search engine, in *Proc. of the 10th Annual Int. Conf. on World Wide Web*, ACM Press, Hong Kong, China, 2001, 162.

[112] Wheeler, D.L., Chappey, C., Lash, A.E., Leipe, D.D., Madden, T.L., Schuler, G.D., Tatusova, T.A., and Rapp, B.A., Database Resources of the National Center for Biotechnology Information, *Nuc. Acids Res.*, 28(1):10-14, 2000.

[113] Westbrook, J., Feng, Z., Chen, L., Yang, H., and Berman, H.M. The Protein Data Bank and structural genomics. *Nuc. Acids Res.* 31 (2003): 489-491.

[114] Westbrook, J., Feng, Z., Jain, S., Bhat, T.N., Thanki, N., Ravichandran, V., Gilliland, G.L., Bluhm, W., Weissig, H., Greer, D.S., Bourne, P.E., and Berman, H.M. The Protein Data Bank: unifying the archive. *Nuc. Acids Res.* 30 (2002): 245-248.

[115] Wu, C.H., Huang, H., Yeh, L.S.L., and Barker, W.C. Protein Family Classification and Functional Annotation, *Computational Biology and Chemistry*, 27:37-47, 2003.

[116] Wu, C.H., Yeh, L.-S., Huang, H., Arminski, L., Castro-Alvear, J., Chen, Y., Hu, Z., Kourtesis, P., Ledley, R. S., Suzek, B.E., Vinayaka, C.R., Zhang, J., and Barker, W.C. The Protein Information Resource, *Nuc. Acids Res.*, 31(1):345-347, 2003.

[117] Zipf, G.K., *Human behavior and the principle of least effort*, Addison-Wesely, Cambridge, Massachusetts, 1949.

[118] Zhang, K., Shasha, D., and Wang, J.T.L., Approximate Tree Matching in the Presence of Variable Length Don't Cares, *J. of Algorithms*, Academic Press, 16, 33, 1994.

[119] Zhang, S., Herbert, K.G., and Wang, J.T.L., XML Query by Example, *Int. J. of Computational Intelligence and Applicantions*, Braga, A.D.P, Wang, J.T.L, Eds., World Scientific Publishing, 2, 329.

[120] Zhu, B., Leroy, G., Chen, H., Chen, Y.: MedTextus: an intelligent web-based medical meta-search system. JCDL 2002: 386.