Correlation-Based Methods for Biological Data Cleaning

JUDICE, LIE YONG KOH
(Master of Technology, NUS)

A THESIS SUBMITTED
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY
DEPARTMENT OF COMPUTER SCIENCE
SCHOOL OF COMPUTING
NATIONAL UNIVERSITY OF SINGAPORE

2007

In long memory of my father and sister

Correlation-Based Methods for Biological Data Cleaning

by

JUDICE, LIE YONG KOH, M.Tech

Dissertation

Presented to the Faculty of

the School of Computing of

the National University of Singapore

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

National University of Singapore

March 2007

Acknowledgements

I would like to express my gratitude to all those who have helped me complete this PhD thesis. First, I am deeply grateful to my supervisor, Dr. Mong Li Lee, School of Computing, National University of Singapore, for her guidance and teachings. This completion of the PhD thesis will not be possible without her consistent support and patience, as well as her wisdom which has been of utmost value to the project.

I would also like extend my gratitude to my mentor, Associate Prof Wynne Hsu, School of Computing, National University of Singapore, for her guidance and knowledge. I am fortunate to have learned from her, and have been greatly inspired by her wide knowledge and intelligence.

I have furthermore to thank my other mentor, Dr. Vladimir Brusic, University of Queensland for providing biological perspectives to the project. And my appreciation goes to the advisory committee members for beneficial discussions during my Qualifying and Thesis Proposal examinations.

In addition, I wish to extend my appreciation to my colleagues in the Institute for Infocomm Research (I^{2}R) for their assistance, suggestions and friendship during the course of my part-time PhD studies. Special acknowledgement goes to Mr. Wee Tiong Ang and Ms. Veeramani Anitha, Research Engineer for their helps and to Dr. See Kiong Ng, Manager of Knowledge Discovery Department for his understanding and encouragement.

Most importantly, I will like to thank my family for their love. I will also like to dedicate this thesis to my sister whose passing had driven me to retrospect my goals in life and to my father who died of heart attack and kidney failure in the midst of my study and whom I regretted for not spending enough time with during his last days. And to the one I respect most in life, my mother.

Last but not least, I wish to express my greatest appreciation to my husband, Soon Heng Tan for his continuous support, encouragement and for providing his biological perspectives to the project. I am thankful that I can always rely on his understanding to help me through the most difficult times of the PhD study and of my life.

Judice L.Y. Koh

National University of Singapore

December 2006

Abstracts

Data overload combine with widespread use of automated large-scale analysis and mining result in a rapid depreciation of the World's data quality. Data cleaning is an emerging domain that aims at improving data quality through the detection and elimination of data artifacts. These data artifacts comprise of errors, discrepancies, redundancies, ambiguities, and incompleteness that hamper the efficacy of analysis or data mining.

Despite the importance, data cleaning remains neglected in certain knowledge-driven domains. One such example is Bioinformatics; biological data are often used uncritically without considering the errors or noises contained within, and research on both the "causes" of data artifacts and the corresponding data cleaning remedies are lacking. In this thesis, we conduct the an in-depth study of what constitutes data artifacts in real-world biological databases. To the best of our knowledge, this is the first complete investigation of the data quality factors in biological data.The result of our study indicates that biological data quality problem is by nature multifactorial and requires a number of different data cleaning approaches. While some existing data cleaning methods are directly applicable to certain artifacts, others such as annotation errors and multiple duplicate relations have not been studied. This provides the inspirations for us to devise new data cleaning methods.

Current data cleaning approaches derive observations of data artifacts from the values of independent attributes and records. On the other hand, the correlation patterns between the attributes provide additional information of the relationships embedded within a data set among the entities. In this thesis, we exploit the correlations between data entities to identify data artifacts that existing data cleaning methods fall short of addressing. We propose 3 novel data cleaning methods for detecting outliers and duplicates, and further apply them to real-world biological data as proof-of-concepts.

Traditional outlier detection approaches rely on the rarity of the target attribute or records. While rarity may be a good measure for class outliers, for attribute outliers, rarity may not equate abnormality. The ODDS (**O**utlier **D**etection from **D**ata **S**ubspaces) method utilizes deviating correlation patterns for the identification of common yet abnormal attributes. Experimental validation shows that it can achieve an accuracy of up to 88%.

The ODDS method is further extended to XODDS, an outlier detection method for semi-structured data models such as XML which is rapidly emerging as a new standard for data representation and exchange on the World Wide Web (WWW). In XODDS, we leverage on the hierarchical structure of the XML to provide addition context information enabling knowledge-based data cleaning. Experimental validation shows that the contextual information in XODDS elevates both efficiency and the effectiveness of detecting outliers.

Traditional duplicate detection methods regard duplicate relation as a boolean property. Moreover, different types of duplicates exists, some of which cannot be trivially merged. Our third contribution, the correlation-based duplicate detection method induced rules from associations between attributes in order to identify different types of duplicates.

Correlation-based methods aimed at resolving data cleaning problems are conceptually new. This thesis demonstrates they are effective in addressing some data artifacts that cannot be tackled by existing data cleaning techniques, with evidence of practical applications to real-world biological databases.

List of Tables

List of Figures

Table of Contents

Chapter 1: Introduction

The beginning of knowledge is the discovery of something we do not understand.

Frank Herbert
US science fiction novelist (1920 - 1986)

1.1 Background

1.1.1 Data Explosion, Data Mining, and Data Cleaning

The "How much information" project conducted by UC Berkeley in 2003 estimated that every year, one person produces an equivalence of "30-feet books" of data, and 92 percent are in electronic formats [LV03]. However, this astonishing quantitative growth of data is the antithesis of its qualitative content. Increasingly diversified sources of data combined with the lack of quality control mechanisms result in the depreciation of the World's data quality - a phenomenon commonly known as *data overloading*.

The first decade of the 21st century also witness a widespread use of data mining techniques that aim at extracting new knowledge (concepts, patterns, or explanations, among others) from the data stored in databases, also known as Knowledge Discovery from Databases (KDD). The prevalent popularity of data mining is driven by technological advancements that generate voluminous data, which can no longer be manually inspected and analysed. For example, in the biological domain, the invention of high-throughput sequencing techniques enables the deciphering of genomes that accumulate massively into the biological databanks. GenBank, the public repository of DNA sequences build and supported by the US National Institute of Health (NIH) has been growing exponentially towards 100 billion bases, the equivalence of more than 70 million database records (Figure 1.1). Similar growth of DNA data are seen in DNA databank of Japan (DDBJ) and European Molecular Biology Laboratory (EMBL). The data available from GenBank, DDBJ and EMBL are only parts of the "ocean" of public-domain biological information which is used extensively in Bioinformatics for *In silico* discoveries – biological discoveries using computer modelling or computer simulations.

Due to the sheer volume, databases such as GenBank are often used with no consideration of the errors and defects contained within. When subject to automated data mining and analysis, these "dirty data" may produce highly misleading results, resulting in a "*garbage-in garbage-out*" situation. Further complication arises when some of the erroneous results are added back into the information systems, and therefore creating a chain of error proliferations.

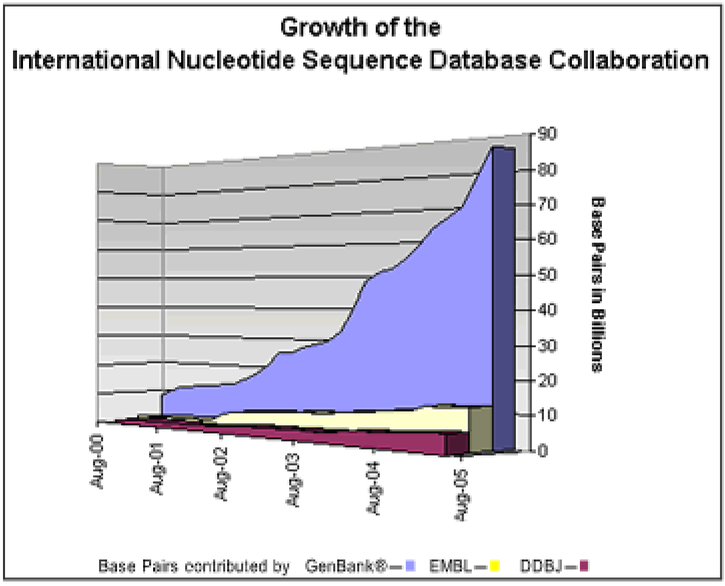


Figure 1.1: Exponential growth of DNA records in GenBank, DDBJ and EMBL

Figure from http://www.ncbi.nlm.nih.gov/Genbank

Data cleaning is an emerging domain that aims at improving data quality. It is particularly critical in databases with high evolutionary nature such as the biological databases and data warehouses new data generated from the worldwide experimental labs are directly submitted into these databases on daily basis without adequate data cleaning steps and quality checks. The "dirty data" accumulate as well as proliferate as the data exchange among the databases and transform through data mining pipelines.

Although data cleaning is the essential first step in the data mining process, it is often neglected conveniently because the solution towards attaining high quality data is non-obvious. Development of data cleaning techniques is at its infancy and the problem is complicated by the multiplicity as well as the complexity of *data artifacts*, also known as "dirty data" or data noise.

1.1.2 Applications Demanding “Clean Data”

High quality data or “clean data” are essential to almost any information system that requires accurate analysis of large amount of real-world data. In these applications, automatic data corrections are achieved through data cleaning methods and frameworks, some forming the key components of the data integration process (e.g. data warehouses) and are the pre-steps of even using the data (e.g. customer or patient matching). This section describes some of the key applications of data cleaning.

1.1.2.1 Data Warehouses

The classical application of data cleaning is in data warehouses [LLLK99, VVS$^+$00, RH01, ACG02, CGGM03]. Data warehousing emerged as the solution for “warehousing of information” in the 1990s in the business domain; a business data warehouse is defined as a subject-oriented, integrated, non-volatile, time-variant collection of data organised to support management decisions [Inm93]. Common applications of data warehousing include:

- Business domain to support business intelligence and decision making [Poe96, AIRR99]
- Chemo-Informatics to facilitate pharmaceutical discoveries [Heu99]
- Healthcare to support analysis of medical data warehouses [Sch98, Gib99, HRM00]

Data warehouses are generally used to provide analytical results from multi-dimensional data through effective summarization and processing of segments of source data relevant to the specific analyses. Business data warehouses are basis of decision support systems (DSS) that provide analytical results to managers so that they can analyse a situation and make important business decisions. Cleanliness and integrity of the data contributes to the accuracy and correctness of these results and hence affects the impact of any decision or conclusion drawn, with direct cost amounting to 5 million dollars for a corporate with a customer base of a million [Kim96].

Nevertheless, the quality problem of data warehouses is non-trivial. In a data warehouse, analytical results are derived from large volume of historical and operational data integrated from heterogeneous sources. Warehouse data exist in highly diversified formats and structures, and therefore it is difficult to identify and merge duplicates for purpose of integration. Also, the reliability of the data sources is not always assured when the data collection is voluminous; large amount of data can be deposited into the operational data sources in a batch mode or by data entry without sufficient checking. Given the excessive redundancies and the numerous ways errors can be introduced into a data warehouse, it is not surprising that data cleaning is one of the fast evolving research interests for data warehousing in the 21st century [SSU96].

1.1.2.2 Customer or Patient Matching

Data quality is sometimes defined as *a measurement of the agreement between the data views presented by an information system and that same data in real world* [Orr98]. However, the view presented in a database is often an over-representation of an entity in real world; multiple records in a database represent the same entity or fragmented information of it.

In banking, the manifestation of duplicate customer records incurs direct mailing costs in printing, postage, and mail preparation by sending multiple mails to the same person and same household. In United States alone, $611 billion a year is lost as a result of solely customer data (names and addresses) [Eck02]. Table 1.1 shows an example of the customer matching problem. As shown, the duplication detection problem is a combination of:

- Mis-spellings e.g. “Judy Koh”
- Typographical errors e.g. “Judic Koh” and “S’pre”
- Word transpositions e.g. “2 13 Street East” and “Koh Judice”
- Abbreviations e.g. “SG” and “2 E 13 St”
- Different data types e.g “Two east thirteenth st”
- Different representations e.g country code can be represented as “(65)”, “65-“ or “(065)”

- Change in external policy such as the introduction of an additional digit to Singapore's phone numbers effective from 2005. "65-8748281" becomes "65-68748281".

Table 1.1: Different records in database representing the same customer

	Name	Address	City	State	Zip	Phone
1	J.Koh	2 E 13th Street	Singapore	-	119613	(65) 8748281
2	Judice	2 13 Street East	SG	Singapore	119-613	68748281
3	Koh Judice	2 E thirteenth street	S'pore	S'pore	11961	65-68748281
4	Judy Koh	2 E 13 St	-	SG	119 613	65-8748281
5	Judic Koh	Two east thirteenth st	Toronto	S'pre	-	(065)-8748281

The data cleaning market-place is loaded with solutions for cleaning customer lists and addresses, including i/Lytics GLOBAL by Innovative Systems Inc. (http://business.innovativesystems.com/postal_coding/index.php), Heist Data Cleaning solutions (http://www.heist.co.uk/mailinglistscleaning/), and Dataflux Corporation (http://www.dataflux.com/main.jsp).

The same redundancy problem prevails in healthcare. Mismatching the patients to the correct medical records, or introducing errors to the prescriptions or patients health records can cause disastrous loss of lives. The Committee of Healthcare in America estimated that 44,000 to 98,000 preventable deaths per year are caused by erroneous and poor quality data; one major cause is mistaken identities [KCD99].

1.1.2.3 Integration of information systems or databases

Data cleaning is required whenever databases or information systems need to be integrated, particularly after acquisition or merging of companies. To combine diversified volumes of data from numerous backend databases, often geographically distributed, enormous data cleaning efforts are required to deal with the redundancies, discrepancies and inconsistencies.

In a classical example, the British Ministry of Defence embarked on an $11 million data cleansing project in 1999 to integrate 850 information systems, 3 inventory systems and

15 remote systems. Data cleaning processes conducted over the four years include (1) disambiguation of synonyms and homonyms, (2) duplicate detection and elimination, and (3) error and inconsistency corrections through data profiling. This major data cleaning project is believed to have saved the British Ministry $36 million dollars [Whe04].

In general, data quality issues are critical in domains which require storage of large volume of data, are constantly integrated from diversified sources, and where data analysis and mining plays an important role. One such example is Bioinformatics.

1.1.3 Importance of Data Cleaning in Bioinformatics

Over the past decade, advancement in high-throughput sequencing offers unprecedented opportunities for scientific breakthroughs in fundamental biological research. While genome sequencings of more than 205,000 named organisms aim at elucidating the complexity of biological systems, this is only the beginning of the era of data explosion in biological sciences. Given the development of faster and more affordable genome sequencing technologies, the numerous organisms that have not been studied, and the recent paradigm shift from genotyping to re-sequencing, the number of genome projects is expected to continue at an exponential growth rate into the next decade [Met05]. These genome project initiatives are directly translated into amounting volumes of uncharacterized data which rapidly accumulates into the public biological databases of biological entities such as GenBank [BKL+06], UniProt [WAB+06], PDB [DAB+05], among others .

Public biological databases are essential information resources used daily by biologists around the world for sequence variation studies, comparative genomics and evolution, genome mapping, analysis of specific genes or proteins, molecular bindings and interactions study, and other data mining purposes. The correctness of decisions or conclusions derived from the public data depends on the data quality, which in turn suffers from exponential data growth, increasingly diversified sources, and lack of quality checks. Clearly, the presence of data artifacts directly affects the reliability of biological discoveries. Bork [Bor00] highlighted that poor data quality is the key hurdle that the bioinformatics community has to overcome in order that computational prediction schemes exceed 70% accuracy. Informatics burdens created by low quality, unreliable data also limits large-scale analysis at the *–omics* (Genomics, Proteomics, Immunomics, Interactomics, among others) level. As a result, the complete knowledge of biological systems remains buried within the biological databases.

Although this need is drawing increasing attention over the last few years, progress still fall short in making the data "fit for analysis" [MNF03, GAD02], and data quality problems of varying complexities exist [BB96, BK98, Bre99, Bor00, GAD$^+$02], some of which cannot be resolved given the limitations of existing data cleaning approaches.

1.1.4 Correlation-based Data Cleaning Approaches

Current data cleaning approaches derive observations of data artifacts from independent attributes and records (details in Chapter 2). On the other hand, the *correlation patterns* embedded within a data set provide additional information of the semantic relationships among the entities, beyond the individual attribute values. Correlation mining - the analysis of the relationships among attributes is becoming an essential task in data mining processes. For example, the core of *association rule mining* is to find sets of attributes that *co-occur* frequently in a transaction database, while *feature selection* involves identifying strongly correlated dimensions.

Table 1.2: Customer bank accounts with personal information and monthly transactional averages

Ac	Type	Cust/ Age	Cust/ Profession	Addr/ Country	Addr/ State	Addr/ City	Trans/ Count	Trans/ Avg
1	Saving	35	Engineer	Czech	S.Moravi	Opava	2	$52
2	Cheque	75	Manager	USA	LA	California	300	$143
3	Saving	16	Professor	Czech	S.Moravi	Opava	80	$72
4	Saving	18	Student	USA	S.Moravi	Opava	58	$63
5	Saving	37	Professor	Czech	S.Moravi	Opava	25	$124

Table 1.2 shows a simple example of the inadequacy of merely considering data values in outlier detection. By applying traditional mechanisms for attribute outlier detection that focus on finding rare values across univariate distributions of each dimension, we may be able to identify the low transaction count in Account 1 is an *attribute outliers*. However, such strategies based on rarity are unlikely to determine the 16-year old professor in Account 3, or the USA that is erroneously associated with the city and state of Czech in Account 4. These possible errors are however detectable from the deviating co-occurrence patterns of the attributes.

Besides *abnormal correlations* that constitute data noise in the form of attribute outliers, the mining of *positive correlations* also enables the sub-grouping of redundancy relations. Duplicate detection strategies typically compute the degree of field similarities between two records in order to determine the extent of duplication. Moreover, intuitively, duplicate relation is not a boolean property because not all similar records can be trivially merged. The different types of duplicates do not vary in their extent of similarity but rather in their associative attributes and corresponding similarity thresholds.

Correlation mining techniques generally focus on strong positive correlations [AIS93, LKCH03, BMS97, KCN06]. Besides market basket analysis, correlation-based methods have been developed for complex matching of web query interface [HCH04], network management [GH97], music classification [PWL01], among others. However, correlation-based methods targeted at resolving data cleaning problems are conceptually new.

1.1.5 Scope of Data Cleaning

Juron and Blanton defined in [JB99] - *"data to be of high quality if they are fit for their intended uses in operations, decision making and planning."* According to this definition, data quality is measured by the usability of data, and achieving high quality data encompasses the definition and management of processes that create, store, move, manipulate, process and use data in a system [WKM93, WSF95]. While a wide range of issues relate to data usability - from typical quality criterion such as data consistency, correctness, relevance to application and human aspects such as ease-of-use, timeliness, and accessibility, current approaches in data cleaning mainly arises out of the need to mine and to analyse large volume of data residing in databases or data warehouses.

Specifically, the data cleaning approaches mentioned in this work devote to data quality problems that hamper the efficacy of analysis or data mining and are identifiable completely or partially through computer algorithms and methods. The data cleaning research covered in this work does not take into account data quality issues associated with the external domain-dependent and process-dependent factors that affect how data are produced, processed and physically passed around. It does not include quality control initiatives, such as manual selection of input data, manual tracing of data entry sources, feedback mechanisms in the data processing steps, the usability aspects of the database application interfaces, and other domain specific objectives associated with the non-computational correction of data.

While we will not give details, it suffices to mention that the term data cleaning has different meanings in various domains; some examples are found in [RAMC97, BZSH99, VCEK05]. For biological data, this work does not cover sequencing errors caused by a defective transformation of the fluorescent signal intensities produced by an automated sequencing machine into a sequence of the four bases of DNA. Such measurement errors are not traceable from the sequence records using statistical computation or data mining.

1.2 Motivation

Data cleaning is an important aspect of bioinformatics. However, biological data are often used uncritically without considering the errors or noises contained within. Relevant research on both the "causes" and the corresponding data cleaning remedies are lacking. This research is driven by the desire to address the data quality problems in real-world data such as the biological data. The thesis has two main objectives:

(1) Investigate factors causing depreciating data quality in the biological data

(2) Devise new data cleaning methods for data artifacts that cannot be resolved using existing data cleaning techniques

To the best of our knowledge, this is the first serious work in biological data cleaning. The benefit of addressing data cleaning issues in biological data is two-fold. While the high dimensionality and complexity of biological data depicts it as an excellent real-world case study for developing data cleaning techniques, biological data also contain an assortment of data quality issues providing new insights to data cleaning problems.

1.3 Contribution

This thesis presents a complete study of the classification of data artifacts in biological databases and proposes 3 new correlation-based data cleaning methods. The classification of biological data artifacts serves as a “roadmap” for data cleaning processes. The data cleaning methods are general; and we demonstrate they are applicable to both biological and non-biological data. These methods are unlike traditional data cleaning strategies that focused on the defects in individual records or attribute values. Rather, the correlations between data entities are exploited to identify artifacts that existing data cleaning methods cannot detect.

The completion of this research project will make 4 specific contributions to the research in data cleaning as well as bioinformatics:

- **Classification of biological data artifacts**

  The data quality problem of biological data is a collective result of artifacts at the field, record, single and multiple-database levels (physical classification), and a combinatory problem of the bioinformatics that deals with the syntax and semantics of data collection, annotation, and storage, as well as the complexity of biological data (conceptual classification). We conduct an investigation to determine the multiple types of data artifacts that cause data quality depreciation in major biological databases; 11 types and 28 subtypes of data artifacts are identified. We classify these artifacts into their physical as well as conceptual types. We also evaluate the limitations of existing data cleaning methods in addressing each type of artifacts. To the best of our knowledge, this is the first complete study of biological data artifacts, with the objective of gaining holistic insights into the data quality problem and the adequacy of current data cleaning techniques.

- **A correlation-based attribute outlier detection method**

  An outlier is an object that does not conform to the normal behaviour of the data set. Existing outlier detection methods focus on class outliers and research on attribute outliers is limited, despite the equal role attribute outliers play in depreciating data quality and reducing data mining accuracy. We introduce ODDS (for **O**utlier **D**etection from **D**ata **S**ubspaces) method to detect attribute outliers from the deviating correlation behaviour of attributes. Three metrics to evaluate outlier-ness of attributes, and an adaptive factor to distinguish outliers from non-outliers are proposed. Evaluation on both biological and non-biological data shows that ODDS is effective in identifying attribute outliers, and detecting erroneous annotations in protein databases.

- **A framework for detecting attribute outliers in XML**

  Increasing biological databases are converted into XML formats in order to facilitate data exchange. However, current outlier detection methods for relational data models are not directly adaptable to XML documents. We develop a novel outlier detection method for XML data models call XODDS (for **X**ML **O**utlier **D**etection from **D**ata **S**ubspace). The XODDS framework utilizes the correlation between attributes to adaptively identify outliers and leverages on the hierarchical structure of XML to determine semantically meaningful subspaces of the correlation-based outliers. XODDS consists of four key steps: (1) attribute aggregation defines summarizing elements in the hierarchical XML structures, (2) subspace identification determines contextually informative neighbourhoods for outlier detection, (3) outlier scoring computes the extent of outlier-ness using correlation-based metrics, and (4) outlier identification adaptively determine the optimal thresholds distinguishing the outliers from non-outliers.

- **An association mining method to detect multiple types of duplicates**

  This work examines the extent of redundancy in biological data and proposes a method for detecting the different types of duplicates in biological data. Duplicate relations in a real-world biological dataset are induced using association mining. Evaluation of our method on a real-world dataset shows that our duplicate rules can accurately identify up to 96.8% of the duplicates in the dataset.

The classification of biological data artifacts was published in *ICDT 2005 Workshop on Database Issues in Biological Database (DBiBD)*. The paper describing the ODDS outlier detection method has been accepted for publication in *DASFAA 2007* [KLHL07], and the XODDS method paper has been submitted [KLHA07]. A full paper on duplicate detection using association mining was published in *ECML/PKDD 2004 Workshop on Data Mining and Text Mining for Bioinformatics* [KLK$^+$04].

1.4 Organisation

The rest of this thesis is organized as follows. First, Chapter 2 reviews current approaches to data cleaning in detail. Background information on bioinformatics and biological database, and the taxonomy of biological data artifacts is presented in Chapter 3. The ODDS method is presented in Chapter 4. We demonstrate how ODDS can be applied to distinguish erroneous annotations in protein databases. An extension of the outlier detection framework to XML data is proposed to Chapter 5, which leverages on the contextual information in XML to facilitate the detection of outliers in semi-structured data models. Chapter 6 presents a correlation-based approach towards duplicate detection of protein sequences. We conclude in Chapter 7 with discussions on further works.

Chapter 2: A Survey on Data Cleaning Approaches

If I have seen further, it is by standing on the shoulders of giants.

Issac Newton
English Mathematician (1643-1727)

In this chapter, we discuss how data cleaning approaches have evolved over the last decade and we survey existing data cleaning methods, systems and commercial applications.

2.1 Data Artifacts and Data Cleaning

Data cleaning, also known as data cleansing or data scrubbing encompasses methods and algorithms that deal with artifacts in data. We formally define data cleaning:

Data cleaning is the process of detecting and eliminating data artifacts in order to improve the quality of data for analysis and mining.

Here, data artifacts refer to data quality problems such as *errors, discrepancies, redundancies, ambiguities,* and *incompleteness* that hamper the efficacy of analysis or data mining. Since real-world objects that are completely and accurately represented in databases have perfect data quality [Orr98], data artifacts are basically the differences between the real-world and database representations. Data artifacts may be caused by erroneous entry, wrong measurements, data transformation problems, inaccurate annotations, mis-interpretations, among others. Table 2.1 shows some common examples of data artifacts and their types.

Table 2.1: Different types of data artifacts

	Errors	Discrepancies	Incompleteness	Redundancies	Ambiguities
Duplicates				*	
Outliers	*	*			
Spurious links				*	
Missing values			*		
Illegal values	*				
Synonyms				*	*
Homonyms					*
Integrity violations	*	*			*
Dependency violations	*				
Format variations					*
Word transposition					*
Mis-spellings	*				*

We broadly characterized data artifacts into 5 types – errors, discrepancies, incompleteness, redundancies and ambiguities. Errors are measurements, observations, or calculations which are incorrect or inaccurate representations of the "truth". In databases, errors are seen as outliers, illegal values, integrity and dependency violations, or mis-spellings. For instance, consider a relation *R*(Country, State, City) and let r_1 = <'Singapore', 'Singapore', 'Toronto'> be a tuple in *R*, where r_1[City]=<'Toronto'> is erroneously introduced. If the functional dependency FD: City → Country is specified in the relational database, it is possible to detect the error as a dependency violation at point of insertion. However, this FD does not always hold. For example, the city called Geneva is in Illinois, U.S.A as well as Switzerland, Geneva (state). An alternative approach is to take into account the deviating behaviour of the attribute and utilize outlier detection approaches to isolate the error.

Discrepancies are differences between conflicting observations, measurements or calculations. Unlike errors, it is not straightforward to determine which of the conflicting entities is the "truth". Consider another tuple in *R*, r_2 = <'Canada', 'British Columbia', 'Toronto'>. r_2[State]=<'British Columbia'> and r_2[City]=<'Toronto'> are conflicting observations because either may be erroneous. Similarly, homonymous entities are not necessary incorrect.

Incompleteness means the information of a real-world entity is missing from the corresponding tuples in the databases. When a highly sparse database, which is manifested with missing values, is subjected to machine learning, the learned model may adjust to very specific random features in the rare training examples, thus resulting in over-fitting. Likewise, in data mining perspectives, redundancy in duplicate or synonymous records results in over-representations of specific patterns that in turn, disturb the statistical distributions.

Ambiguities refer to unclear or uncertain observations. The use of multiple names to describe the same entity (synonyms), the same names for different entities (homonyms), or mis-spellings are all symbolic of ambiguous information. For example, besides known as a common abbreviation for two different classes of enzymes - glycerol kinase and guanylate kinase, GK is also as an abbreviation of the Geko gene of *Drosophila melanogaster* (Fruit fly). It is impossible to tell from the name GK, if the corresponding DNA sequence is an enzyme or is a gene of fruit fly.

Some of these artifacts can be trivially resolved using proprietary spell-checkers and by incorporating integrity, dependency and format constraints into the relational databases. On the contrary, detecting and eliminating duplicates and outliers have proven to be of greater challenge. In fact, majority of the research in data cleaning are related to either duplicate or outlier detection. The alternative approach of hand-correcting the data is extremely expensive and laborious and cannot be fool-proofed of additional entry errors from the annotators. On the other hand, data cleaning is more than a simple update of a record, often requiring decomposition and reassembling of the data. A serious data cleaning tool can easily be an extensive software system.

2.2 Evolution of Data Cleaning Approaches

Data cleaning is a new field that emerges over the last decade. Driven by information overload, widespread use of data mining and developments in database technologies, the data cleaning field has expanded in many aspects; new types of data artifacts are addressed, more sophisticated data cleaning solutions are available, and new data models are explored.

The first works in data cleaning focused at detecting redundancies in data sets (merge/purge and duplicate detection), addressing various types of violations (integrity, dependency, format violations), and identifying defective attribute values (data profiling). Recent works have expanded beyond the defects in individual records or attribute values into the detection of defective relationships between records and between attributes (spurious links). Also, the technical aspects have advanced from individual algorithms and metrics (sorted neighbourhood methods, field matching) into complete data cleaning systems and frameworks (IntelliClean, Potter's wheel, AJAX), as well as essential components of the data warehouse integration systems (ETL and fuzzy duplicates). The data models investigated extend from structured (relational) data to the semi-structured XML models (DogmatiX).

In this thesis, we expand the scope of data cleaning beyond the defects in individual records or attribute values into the detection of defective relationships between records and between attributes. We also delve into data cleaning methods for XML models.

2.3 Data Cleaning Approaches

Strategies for data cleaning may differ according to the types of data artifacts, but they generally faced the *recall-precision dilemma*. We first define recall and precision using true-positives (TP), false-positives (FP) and false-negatives (FN).

$$precision = \frac{TP}{(TP + FP)}$$

$$recall = \frac{TP}{(TP + FN)}$$

Precision, also known as *positive predictive value* is the ratio of data points detected that indeed contain artifacts. Recall, also known as *sensitivity* is the ratio of data artifacts detected. In this work, we use F-score which is a combined score of both recall and precision.

$$F - score = \frac{(2 \times precision \times recall)}{(precision + recall)}$$

The recall-precision dilemma indicates that the higher the recall, the lower is the precision, and vice versa. Data artifacts detection methods are commonly associated with criteria or thresholds that differentiate the artifacts from the non-artifacts. Higher recall can be achieved by relaxing some of the criteria or thresholds with an increase in the number of TP, but corresponding reduction in precision because FP also increases. Stringent criteria or high thresholds may reduce FP and thus increase precision, but at the same time, reduces the number of positive detected and thus the recall. Achieving both high recall and precision, and therefore a high F-score is a common objective for data cleaners.

2.3.1 Duplicate Detection Methods

Early works in data cleaning focused on the *merge/purge* problems, also known as *de-duplication*, *de-duping*, *record linkages, duplicate detection*. Merge/Purge addresses the fundamental issue of *inexact duplicates* – two or more records of varying formats and structures (syntactic values) are alternative representations of the same semantic entity [HS95, HS98]. Merge refers to the joining of information from heterogeneous sources and purge means the extraction of knowledge from the merge data. Merge/purge research generally address two issues:

- **Efficiency** of comparing every possible pair of records from a plurality of databases. The naïve approach has a quadratic complexity, so this class of methods aim at reducing time complexity through restricting the comparisons to records which have higher probability of being duplicates.
- **Accuracy** of the similarity measurements between two or more records. Methods belonging to this class investigate the various similarity functions of fields, especially of strings and multiple ways of record matching.

Duplicates are common in real-world data that are collected from external sources such as through surveying, submission, and data entry. Integration of databases or information systems also generates redundancies. For example, merging all the records in Table 2.2 requires identifying that “First Name” and “Given name” refer to the same entities, “Name” is a concatenation of first and last names, and “Residential” and “Address” refers to the same fields.

Table 2.2: Different records from multiple databases representing the same customer

	Name	Address	City	State	Zip	Phone
1	J.Koh	2 E 13th Street	Singapore	-	119613	(65) 8748281
2	Koh Judice	2 13 Street East	SG	Singapore	119-613	68748281

	First Name	Last Name	Address	Country code	Contact
1	J.	Koh	2 E 13th Street, Singapore	65	8748281
2	Judy	Koh	2 E 13 St. S(119613)	-	68748281

	Given name	Last name	Residential	Country	Tel
1	Judic Koh	Two east thirteenth st	SG	-	(065)-8748281

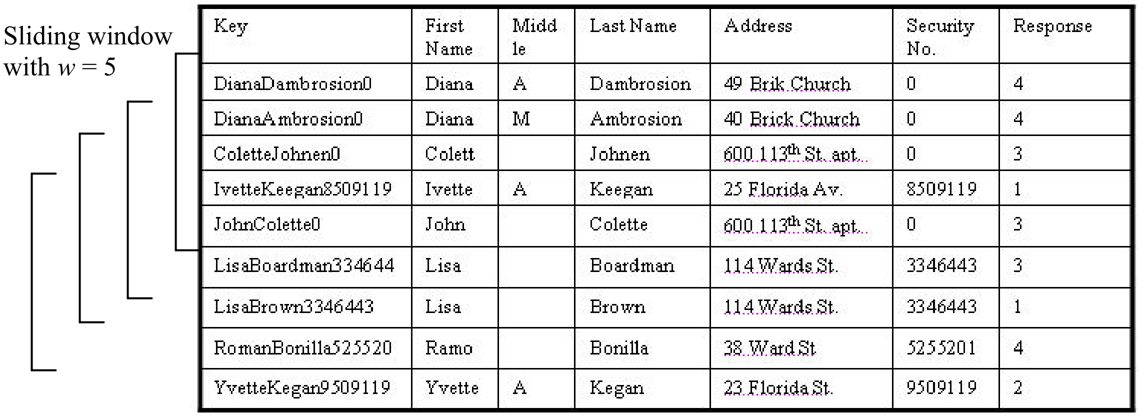
In data warehouses designed for On-Line Analytical Processing (OLAP), Merge/Purge is also a critical step in the Extraction, Transformation, and Loading (ETL) process of integrating data from multiple operational sources.

2.3.1.1 Efficiency-driven Methods

Sorted-neighbourhood method (SNM) is one of the classical approaches to merge/purge problems. SNM first sorts the database based on a unique composite key constructed from one or more fields in order to bring similar records to a bounded neighbourhood in a linear list [HS95]. A window of size *w* is slide along the list such that only records within the window are pair-wise compared; every new record entering the window is compared with the previous *w*-1 records (Figure 2.1). SNM reduces $O(N^2)$ complexity of a typical pair-wise comparison step to $O(wN)$ where *w* is the size of the window and *N* is the number of records. The effectiveness of the method, however, is restricted to the selection of appropriate keys.

An example of SNM is given in Figure 2.1, which shows a list of sorted customer portfolios. The composite key is the combination "<First name><Last name><security ID>". Notice that the accuracy of SNM is highly dependent on the choice of the keys as well as the window width. We can bring the duplicate records "IvetteKeegan8509119" and "YvetteKegan9509119" into lexicographical proximity of the sliding window of size w using the composite key "<First name><security ID><Last name>"; "Keegan8509119Ivette" and "Kegan9509119Yvette" are sufficiently close keys. However, this brings "DianaDambrosion0" and "DianaAmbrosion0" – with corresponding new composite keys "Dambrosion0Diana" and "Ambrosion0Diana" beyond comparable range. Enlarging the size of sliding window may improve the recall of SNM but at the expense of time complexity.

The duplicate elimination method (DE-SNM) improves SNM by first sorting the records on a chosen key and then dividing the sorted records into two lists: duplicate and non-duplicate [Her95]. DE-SNM achieves slight efficiency improvement over SNM, but suffers from the same drawbacks as SNM. The multi-pass sorted-neighbourhood method (MP-SNM) removes SNM's dependency on a single composite key by performing multiple independent passes of SNM based on different sorting keys. The union of the duplicates found from multiple passes are flagged as duplicates. Using the same example in Figure 2.1, 3 separate passes of SNM using “First name”, “Last name” and “Security No.” respectively would have identified all duplicates.



Sliding window
with $w = 5$

Key	First Name	Middle	Last Name	Address	Security No.	Response
DianaDambrosion0	Diana	A	Dambrosion	49 Brik Church	0	4
DianaAmbrosion0	Diana	M	Ambrosion	40 Brick Church	0	4
ColetteJohnen0	Colett		Johnen	600 113th St. apt..	0	3
IvetteKeegan8509119	Ivette	A	Keegan	25 Florida Av.	8509119	1
JohnColette0	John		Colette	600 113th St. apt..	0	3
LisaBoardman334644	Lisa		Boardman	114 Wards St.	3346443	3
LisaBrown3346443	Lisa		Brown	114 Wards St.	3346443	1
RomanBonilla525520	Ramo		Bonilla	38 Ward St	5255201	4
YvetteKegan9509119	Yvette	A	Kegan	23 Florida St.	9509119	2

Figure 2.1: Sorted Neighbourhood Method with sliding window of width 6

In [ME97], priority queues of clusters of records facilitate duplicate comparison. Instead of comparing to every other record within a fixed window, a record is compare to representatives of clustered subsets with higher priority in the queue. It reported a saving of 75% of time from the classical pair-wise algorithm.

Transitivity and Transitivity Closure

Under the assumption of transitivity, if record x_1 is a duplicate of x_2, and x_2 is a duplicate of x_3, then x_1 is a duplicate of x_3. Some duplicate detection methods leverage on the assumption that relation “is duplicate of” is transitive to reduce the search space for duplicates [HS95, LLKL99, ME97]. Generalizing the transitivity assumption, we denote $x_i \approx x_j$ if x_i is a detected duplicate of record x_j. Then for any x which is a duplicate of x_i, $x \approx x_j$. Likewise, $x \approx x_j$ implies that $x \approx x_i$. With the transitive assumption of duplicate relations, the number of pair-wise matching that is required to determine clusters of duplicates is reduced.

If we model data records into an undirected graph where edges represent the relation “is similar to”, then the “is duplicate of” relation corresponds to the transitive closure of the “is similar to” relation. Further clarifying, we define formally transitive closure:

Let R be the binary relation "is similar to" and X be a set of duplicate records. The transitive closure of R on a set X is the minimal transitive relation R' on X that contains R.. Thus for any $x_i, x_j \in X$ *,* $x_iR'x_j$ *iff there exist* x_i*,* x_{i+1}*, ...,* x_j *and* x_rRx_{r+1} *for all* $i \leq r < j$*.*

R' is a transitive closure of R means that x_i is reachable from x_j and vice versa. In a database, a transitive closure of "is duplicate of" can be seen as a group of records representing the same semantic entity.

However, the duplicate transitivity assumption is not flawless without loss of precision; the extent of similarity diminishes along the transitive relations. Two records, which are far apart in the "is similar to" graph, are not necessarily duplicates. An example is given in [LLL00]: "Mather" ≈ "Mother" and "Mather" ≈ "Father", but "Mother" ≈ "Father" does not hold.

2.3.1.2 Accuracy-driven methods

Instead of reducing the complexity of pair-wise comparisons, other duplicate detection research focus on the accuracy of the determining duplicates. These works generally relate to record linkages, object identification, and similarity metrics. The duplicate determination stage decomposes into two key steps:

(1) *Field Matching* measures the similarity between corresponding fields in two records.

(2) *Record Matching* measures the similarity of two or more records over some combinations of the individual field matching scores.

Field Matching Functions

Most field matching functions deal with string data types because typographical variations in strings account for a large part of the mismatches in attribute values. A comprehensive description of the general string matching functions is given in [Gus97]. [EIV07] gives a detailed survey of the field matching techniques used for duplicate detection. Here, we will highlight a few commonly used similarity metrics.

String similarity functions are roughly grouped into *order-preserving* and *unordered* techniques. Given that order-preserving similarity metrics rely on the order of the characters to determine similarities; these approaches are suitable for detecting typographical errors and abbreviations.

The most common order-preserving similarity function is the *edit distance*, also known as the Levenshtein distance, which calculates the number of operations needed to transform from one string to another [Lev66]. For example, the edit distance between “Judice” and “Judy” is 3 because 3 edits - 1 substitution and 2 deletions are required for the transformation. The basic algorithm for computing edit distance using dynamic programming (DP) runs at the complexity of $\mathrm{O}(|s_1| \times |s_2|)$ where $|s_1|$ and $|s_2|$ are the lengths of the strings s_1 and s_2 respectively.

Recent years has seen the adaptation of string matching strategies originally used in Bioinformatics to align DNA (string of nucleotides) or protein (string of amino acids) sequences. Unlike edit distance, these sequence similarity functions allow for *open gaps* and *extend gaps* between the characters at certain penalties [NW70, SW81]. For example, edit distance is highly position-specific and does not effectively match mis-aligned string such as “J. L. Y. Koh” with “Judice L. Y. Koh”. With *Needleman and Wunsch algorithm* [NW70] and *Smith-Waterman distance* [SW81], the introduction of gaps into the first string enables proper alignment of the two strings. However, studies had shown that more elaborated matching algorithms such as Smith-Waterman does not necessarily out-performed basic matching functions [BM03].

Unordered string matching approaches do not require the exact ordering of characters and hence are more effective in identifying word transpositions and synonyms. The notion of “token matching” was introduced in [LLKL99]. Tokenizing a string involves 2 steps: (1) Split each string into tokens delimited by punctuation characters or spaces, and (2) Sort the tokens lexicographically and join them into a string which is used as the key for SNM and DE-SNM. It makes sense to tokenize strings semantically because different orderings of real-world string values often refer to the same entity. For example, tokenizing both “Judice L. Y. Koh” and “Koh L. Y. Judice” with different ordering of the first, middle and last names produces "Judice L. Koh Y." as the key for record matching in SNM. Similar concept of "atomic tokens" of words calculates the number of matching tokens from two strings to determine the similarity between 2 fields [ME96].

Another unordered string similarity function is the cosine similarity that transforms the input strings into vector space to determine similarity using the Euclidean cosine rule. Cosine similarity of two strings s_1 and s_2 represented by "bag of words" w is defined

$$\cos ine(s_1, s_2) = \frac{\sum_w s_1(w).s_2(w)}{\sqrt{\sum_w s_1(w)^2.s_2(w)^2}}$$

String similarities can also be machine-learned, using support vector machine (SVM) or probabilistic approaches [BM03]. While learning approaches towards string similarity has the benefit of adapting the algorithm according to different input databases, the accuracy is highly dependent on the size of the input data set, and it is difficult to find training data sets with sufficient coverage of similar strings.

Record Matching Functions

The record matching functions, also known as merging rules determine whether two records are duplicates. A record matching function is defined over some or all of the attributes of the relation. The first record matching methods use simple domain-specific rules specified by domain experts to define a unique collective set of keys for each semantic entity; duplicates of the same object have the same values for these keys [WM89].

In [HS95], merging rules are represented using a set of equational axioms of domain equivalence. For example, the following rule indicates that an identical match of last name and address, together with an almost similar match of last name infer that two records r_i and r_j are duplicates:

Given two records, r_i and r_j
IF the last name of r_i equals the last name of r_j,
AND the first names differ slightly,
AND the address of r_i equals the address of r_j
THEN
r_i is equivalent. to r_j.

A database may require more than one equational axiom to determine all possible duplicate scenarios. Creating and maintaining such domain specific merging rules is time-consuming and is almost unattainable for large databases.

Let S be a general similarity metric of two fields (e.g edit distance) and α be given thresholds. Notice that the above merging rule can be generalized into a conjunction of field similarity measures:

Given two records r_i and r_j, r_i is equivalent to r_j if
$S(r_i[last\ name], r_j[last\ name]) \leq \alpha_1$
$\wedge\ S(r_i[address], r_j[address]) \leq \alpha_2$
$\wedge\ S(r_i[last\ name], r_j[last\ name]) \leq \alpha_3$

Instead of returning a boolean decision of whether r_i and r_j are duplicates, the conjunction can return an aggregate similarity score that determines the extent of replication of the two records [ME97, Coh00]. An alternative method mapped the individual string distances onto a Euclidean space to perform a similarity join [JLM03]. In cases where multiple rules describe the duplication scenarios, the conjunctive clauses are joined disjunctively.

One way to overcome the time-consuming process of manually specifying record matching functions is to derive them through machine learning. The main difficulty in machine learning approaches is the collection of the input training pairs of duplicates and non-duplicates. [SB02] proposed an iterative de-duplication system that actively learns as users interactively label the duplicates and non-duplicates and add them to the classifiers. An accuracy of up to 98% is achievable using Decision Tree C4.5, Support Vector Machine (SVM), and Naïve Bayes as the classifiers. The TAILOR system adopt a supervise classifier approach; the probabilistic, induction, and clustering decision models are used to machine learn the comparison vectors and their corresponding matching or unmatching status [EVE02].

2.3.1.3 Correlation-based Methods

Recent approaches towards duplicate detection utilize context information derived from the correlations of an entity in order to improve the accuracy of matching [ACG02, LHK04]. [ACG02] leverages on the hierarchical correlations between tuples in dimensional tables to detect duplicates with the same correlations across related parent and child tables. [LHK04] exploit the context information of the correlated attributes to determine duplicates. Two records are duplicates if their context attributes overlap significantly. For example, to determine if “Judice L. Y. Koh” and “J. L. Koh” refers to the same author, the spurious link method evaluates the extent of overlap in the contextual information of the co-authors, the subjects, and the concept hierarchies of the conferences or journals where their works are published.

Rather than inspecting individual attributes and records, correlation-based duplicate detection approaches aim at exploiting additional knowledge from the associations between attributes and between the records to improve the efficacy of determining duplicates.

2.3.2 Outlier Detection Methods

An outlier is an object exhibiting alternative behaviour in a data set. It is a data point that does not conform to the general patterns characterizing the data set. Detecting outliers has important applications in data cleaning as well as in the mining of abnormal patterns for fraud detection, stock market analysis, intrusion detection, marketing, network sensors, email spam detection, among others [Esk02, LSM99, PPKG03]. Data cleaning applications depict outliers as data noise or errors interfering with data mining mechanisms and thus, eliminating outliers leads to better accuracy. In other applications, outliers are irregular patterns from the rest of the data and thus entail special notice.

There are two types of outliers, the class and the attribute outliers [ZW04]. *Class outliers* are multivariate data points (tuples) which do not fit into any cluster formed by the remaining data. Intuitively, clustering a data set produces discrimination of class outliers as by-products. *Attribute outliers* are univariate data points deviating from the behaviour of remaining attribute points of the data set.

Existing outlier detection methods have primarily focused on class outliers. Numerous methods for identifying class outliers broadly classifies into distribution-based, clustering-based, distance-based and density-based approaches. Detecting attribute outliers, on the contrary, had received less attention from the data mining community.

2.3.2.1 Distribution-based Approach

Distribution-based approaches are among the first methods designed for detecting outliers. Typically, a distribution model (E.G. Gaussian, Normal) that best-fit the values of an attribute is used to differentiate points which do not fit into the distribution [BL94, RL87]. Although distribution-based methods focus on identifying attribute outliers, the distribution model is usually univariate; they do not take into account correlations between attributes and are limited to the detection of obvious off-scale values in a single dimension. The accuracy largely depends on the best-fit distribution models used, and selecting appropriate models are computationally intensive.

2.3.2.2 Data Polishing

Data polishing approaches to attribute outlier detection problem construct for each dimension a classifier based on the remaining dimensions and the class dimension [Ten04, ZW04]. Incorrect predictions are labelled as attribute outliers. The accuracy of data polishing method varies depending on the classifier used. Generally, classifiers robust to random data noise give better accuracy. In addition, data polishing methods are limited to finding attribute outliers resulting in change of class membership.

2.3.2.3 Clustering-based Approach

Some clustering algorithms generate outliers as by-products, usually in forms of singletons that do not fit into any of the clusters. For example, in [BC00], data points are added to clusters incrementally with the objective of minimizing the change in fractal dimensions. Outliers are thus isolated into the “miniature” clusters. While these methods are optimized to produce clusters rather than outliers, some recent works focused directly on the problem of outlier detection using clustering techniques. [HXD03] utilizes the size of each data cluster and relative distance from neighbouring clusters to compute its outlier-ness. In [JTS01], a Minimum Spanning Tree (MST) is constructed and the clusters at the longest edges are discriminated as outliers. Ren *et al.* [RRP04] optimized the efficiency of clustering-based outlier detection method using a vertical P-tree data representation. In general, clustering-based outlier detection methods have the difficulty to scale with high-dimensional, sparse, and large data sets. The cost of clustering expedite when the data dimensionality and size increases. Aggrawal and Yu developed an evolutionary approach to direct the searches for subspace outliers of lower dimensions [AY05].

2.3.2.4 Density-based Approach

Breunig *et al.* [BKNS00] proposed the first density-based outlier detection method. The number of points in its surrounding neighbourhood defines a density-based outlier. Isolated data points relatively far from its local neighbours are determined by a high local outlier factor (LOF). Jin *et al.* [JTH01] proposed improvement to LOF using pruning of the micro-clusters of compressed data representation in the feature space.

Density-based approaches generally suffer from high computational cost due to the large number of k-nearest neighbour queries. The accuracy again depends on the number k specified as the neighbourhood of the point. The LOCI method reduced the computational cost through approximate calculation using a "box-counting" mechanism [PKGF03]. Still, the speed and accuracy depends on the number of boxes defined.

2.3.2.5 Distance-based Approach

A data point is a distance-based outlier if there exists less than β fraction of other data points which are less than κ distance from it [KNT00]. Native distance-based outlier detection methods do not scale well with data dimensionality and size. Computational time can be reduced by pruning in data partitions [RRK00], p-tree data structures [RRPS04], or distance-based neighbourhood cells [KN98]. The limitation of distance-based approach lies in the selection of β and κ, which are user-defined. Accuracy of the method fluctuates depending on these two parameters. Too high β leads to more false positives while low κ causes more false negatives.

In [KN99], Knorr and Ng mine attribute subspaces in a lattice structure to provide specific explanations to the class outlier identified by their distance-based outlier detection method [KN98]. This approach is restricted to only class outliers but non-class outliers may also contain implicit irregularities in the form of attribute outliers. Whether the presence of attribute outliers constitutes class outliers depends on a number of factors, such as the dimensionality of the data set, the "strength" of the attribute outliers, and the correlation of the outlier attributes and the class attributes.

2.3.3 Other Data Cleaning Methods

Apart from duplicate and outlier detection, data cleaning methods also aimed at resolving artifacts such as dependency, format, and integrity violations, spelling errors, illegal values, and missing values.

2.3.3.1 Fuzzy Matching

Given an initial set of clean records, it is possible to identify new records containing erroneous strings that match fuzzily to those in the existing records [CGGM03]. Consider that a new record <"Usa", "S Diego", "California"> is added to a database relation *R*(Country, State, City). Through fuzzy matching of the individual fields with existing tuples in *R*, it is possible to identify that "Usa" is variant of "U.S.A." and "S. Diego" is a synonym of "San Diego". Fuzzy matching uses the *inverted document functions* (IDF) of the weighted tokens in a record to compute the similarity function, and then identifies *K* nearest neighbours. The method also clusters similar reference records in order to achieve more efficient querying of matching records and fields.

2.3.3.2 Data Profiling

Data profiling techniques derive descriptive metadata features from individual attributes for quality checking of new tuple [RH00]. Some examples of descriptive features include data type, length, cardinality, discrete values, minimum and maximum values, and mean. There is little research interest in data profiling, but it is a common technique used in commercialized data cleaning products due to its simplicity. Examples of commercialized companies using data profiling include data integration solution providers such as Dataflux Corporation (http://www.dataflux.com/main.jsp) and Informatica Corporation (www.informatica.com).

2.3.3.3 Probabilistic Noise Identification

The LENS systems presented in [KM03] identifies corrupted attribute values through the use of 3 probabilistic models of clean and noisy records, and the corruption matrix. The probabilistic models are generated through an iterative process of learning the generative models and estimating the corruption matrix that indicate which attributes are corrupted by noise.

2.3.3.4 Database Repair

The class of research on *database repair* focuses on enforcing integrity constraints to detect and eliminate inconsistencies and conflicts in databases. The process of database repair involves modifying values or deleting tuples in order to ensure that the integrity constraints are satisfied. [BFFR05] models the database repair process into a cost-model of finding the minimal cost repairs to each violation of functional dependencies (FDs) or inclusion dependencies (INDs). While [ADNB06] also regard the repair process as a set of value modifications, it models the problem into a logical theory of signed formulae with the objective of achieving value correction with the least number of modifications. Database repairs are also associated with consistent query answering where repairs are related to the insertion or deletion of records, depending on the queries [Wij05, BC03].

2.4 Data Cleaning Frameworks and Systems

Over the past few years, several data cleaning systems and frameworks become available in the marketplace and for public usage, especially on the Web. These tools provide complete solution towards the data quality problem and typically address more than one artifact.

2.4.1 Knowledge-based Data Cleaning Systems

The IntelliClean is a general knowledge-based data cleaning system [LLL00]. The IntelliClean framework is separated into 3 main stages: (1) Pre-processing steps utilize lookup tables and reference functions to standardized the data format, thus removing variations in spellings, representations, abbreviations, naming, and measurement units. (2) Processing stage performs duplicate detection on the conditioned records using rules specified with an expert system; each rule is weight according to its duplicate detection effectiveness. (3) Validation and verification stage for checking of undetected duplicates and merged results.

By allowing representation of the domain knowledge in the data cleaning framework, IntelliClean achieves both high recall and precision. Though it focuses on duplicate detection, part of the spelling errors, ambiguities in syntactic and semantic representations are resolved at the pre-processing stage.

2.4.2 Declarative Data Cleaning Applications

The Potter's Wheel allows users to "interact" with the data cleaning process of detecting inconsistencies in structures, discrepancies, and errors. Traditional data cleaners either accepts a set of duplicate identification rules (sometimes in form of equational axioms) as inputs to their systems or the rules are hard-coded directly into transformation scripts, usually not by the end-users. Potter's Wheel provides a graphical interface for users to specify the structural transformation process and discrepancy detection rules called domains. The data cleaning process is domain independent because it is entirely user-driven; the system merely facilitates the specifications of the data cleaning steps.

The AJAX approach [GFSS00] models a data cleaning process into a graph of atomic transformations and provides SQL extension for each transformation. It supports user interventions to fine-tune the data cleaning process and to declare the record matching rules. The data lineage facilities makes it possible for users to backtrack steps in the data cleaning process, inspect intermediate results and exceptions. By flagging exceptional cases, the process allows human intervention to handle cases that cannot be automatically resolved.

Also based on declarative data cleaning, ARKTOS provides graphical and declarative features to define the data transformation and cleaning tasks in the data warehouse [VVS$^+$00]. Users using 2 specification languages define the cleaning rules in ARKTOS: (1) XML-based Activity Definition Language (XADL) and (2) Simple Activity Definition language (SADL). The data cleaning rules and tasks model into a pipeline of activities making up the data cleaning process. The system measures the quality of data after each activity using quality factors.

In general, declarative data cleaning applications enable users to model their own data cleaning process. This is facilitated through user-friendly graphical interfaces equipped with declarative features for specifying the rules and tasks. Therefore, such applications are also domain independent.

2.5 From Structured to Semi-structured Data Cleaning

The proliferation of semi-structured data models such as XML, driven by the popularity of the WWW, has created new challenges in data mining. Knowledge discovery activities now encompasses the development of new data mining approaches for more effective and efficient mining of XML data, some of which leverage on the self-describing nature of XML data to provide additional context information to data mining processes [BMBA00, SCH$^+$98].

Unlike other research in data mining, current works in data cleaning primarily focus on structured relational databases and are not applicable or easily extensible to semi-structured data such as XML. There exist limitations in direct adaptation of data cleaning methods for relational data models onto XML data models, given the intrinsic differences between XML and relational data models. XML data is hierarchical, but relational data has a flat structure. XML data is self-describing and has an inherent ordering. These unique characteristics of the XML data models give rise to context information lacking in the relational data model. For instance, an XML document contains information about the relationships of entities to one another in the form of the hierarchy; the only types of relationships that can be defined in relational tables are the parent and dependent table.

This section briefly describes the few known research works on XML data cleaning.

2.5.1 XML Duplicate Detection

DogmatiX is a duplicate detection method for XML data models [WN05]. Unlike the relational counterparts, DogmatiX performs similarity matching of both the data values as well as the structures of the target entities. The method addresses two main difficulties when detecting duplicates in XML documents:

(1) In XML, the descriptions of an object are distributed in different elements which are not necessarily the children. A child element may be a related object. For example, an XML element <Movie> has elements <Title> and <Actors> which is a nested structure of <Actor>. While <Title> element is a feature of the movie, <Actors> refer to a different object which would have been modelled as a separate table in relational model.

(2) Structural diversity of determining the same object in different XML structures. In the last example, the <Actor> elements can also be the direct children of <Movie> element.

To address (1), the descriptive elements of an object are defined heuristically, such as its r-distant ancestors or descendents. Structural diversity is tackled through the use of object definitions that may be provided by the users, to map corresponding objects from different XML structures.

2.5.2 Knowledge-based XML Data Cleaning

[LTLL03] investigates the numerous limitations of conducting knowledge-based duplicate detection based on expert system on XML documents. Three key issues are identified: (1) Mapping to an expert system fact template requires well-defined, ordered columns but XML documents are inherently semi-structured and often contain nested relations. (2) Efficiency of the data cleaning process is dependent on the parser used. (3) Sorting of XML documents by keys used for detecting duplicates in SNM requires mechanism to index the records for manipulation, which is achievable using through conversion of the XML document to a RDBMs or a flat file.

Another work related to cleaning XML documents is [YLL03]. A “template-based” approach is used to select relevant tags from XML web pages to construct a style-tree (ST), with the purpose of eliminating nodes not specified in the style-tree

2.6 Biological Data Cleaning

While the problem of data artifacts in biological data has been known for a long time and individual artifacts have been reported [OH98, Bre99, BC01, GAD$^+$02, ITA$^+$03, MNF03, SFM$^+$99, GAA$^+$00, LKSV92, SS03, Tha99, PHBR04], the development of data cleaning approaches in the bioinformatics domain is at its infancy. Very few complete data cleaning methods for biological data exist.

2.6.1 BIO-AJAX

The BIO-AJAX tool for detecting and resolving duplicate taxonomy of organisms utilize prefix-matching strategies to integrate different terms that describe the same species [HGP$^+$04]. BIO-AJAX uses a list of prefixes as matching keys to gather all trees refer to the same organism. For example, it recognized that the term “homo sapiens” and “homo sapien” refers to the same organism (human), given that one is a prefix of another. BIO-AJAX is integrate into Treebase (www.treebase.org), a database of published phylogenetic trees and related references. As the name implies, the BIO-AJAX is built upon the AJAX data cleaning system for declarative definitions of the matching rules.

2.6.2 Classifier-based Cleaning of Sequences

A case study of handling noises in Osteogenesis Imperfecta (OI) related sequences is present in [Ten03]. Teng applied three data cleaning approaches to resolve the presence of misclassified instances. The first approach utilizes the inherent feature in a C4.5 classifier to avoid over-fitting. The second approach removes incorrect instance predictions (filtering), and the third method correct instances with its predicted values (polishing). Through these cleaning approaches, the classification accuracies improve up to 66%.

2.7 Concluding Remarks

Majority of data cleaning methods focus on the more challenging duplicate and outlier detection problems, while other approaches address database repair issues related to various types of violations, inconsistency, and errors. The data cleaning methods propose in this thesis also target attribute and outlier detection. Instead of identifying defects in individual records or attribute values, we leverage on the correlations embedded within the data set to devise effective methods to identify data artifacts.

For many domains that involve the analysis and knowledge extraction of large volume of data, data quality is an important factor to the accuracy of the analysis and data mining methods. Data cleaning is a critical pre-step for improving data quality and it involves a wide spectrum of approaches for each type of artifacts. Bioinformatics has the same demand for high quality data, but there are limited data cleaning applications in the domain. In the first place, there is little understanding of what causes the data quality problem in biological data. For this reason, it is essential to first conduct a study of the different types of artifacts in biological data.

Chapter 3: A Classification of Biological Data Artifacts

The most incomprehensible thing about the world is that it is comprehensible.

Albert Einstein
Physicist (1879 - 1955)

Bioinformatics is a field where data grows at an exponential rate and knowledge grows only at linear rate. The deciphering of the human genomes and of many more organisms bring about a quantitative data growth that is inverse compare to its qualitative content. Increasing data artifacts depreciate the quality of biological data, and affect large-scale *–omics* analysis. In order to address the data quality problems in Bioinformatics, we must first understand what constitute data artifacts in biological data and the sources of these data quality factors. In the first part of this chapter, we give a brief description of the background of biological data and databases, and the role data cleaning plays in biological database systems. In the second part, we present the result of our investigation of factors that causes depreciation of biological data quality. Through observations derived from biological databases, we identify 11 types and 28 subtypes of biological data artifacts and classify them into their physical and conceptual groupings. Based on heuristics and the domain knowledge, we develop programs to detect these artifacts in representative data sets in order to evaluate the extent of manifestation in real-world databases. For each data artifacts, we identify the appropriate data cleaning methods.

The classification of biological data artifacts serves as a “roadmap” for data cleaning. Mapping the classification to existing data cleaning methods reveals some data artifacts that current data cleaning methods fall short of addressing. To the best of our knowledge, this is the first complete study of the data quality issues in biological data.

3.1 Background

3.1.1 Central Dogma of Molecular Biology

Understanding the issues pertaining to biological data artifacts demands the biological perspectives. At the core of the biological information system is the *central dogma of molecular biology* that depicts the information flow among the real-world biological entities (Figure 3.1). It summarizes the process of “DNA makes RNA makes protein" [Cri58].

Deoxyribonucleic acid (*DNA*) is the hereditary information found in the cell nucleus of human or almost all other organisms. A polymer or chain of four *nucleotides* - Alanine (A), Cytosine (C), Guanine (G) and Thymine (T), a DNA sequence is often represented as a succession of *A*, *C*, *G* and *T* in database records. An important property of DNA is that it replicates; each strand of DNA in the double helix serves as a pattern for replicating the complementary strand. This is critical during cell division so that each cell have the same copy of the DNA as the old cell. While DNA serves as the "blueprint" for hereditary information during cell divisions, it is not directly involved in the biochemical processes of a cell. Rather, *proteins* are the essential functional units (macromolecules) involved in all biochemical processes in living cells. Proteins are made from units of DNA along the chromosomes known as *Genes* through a two-stage process. First, enzymes known as polymerases *transcribed* the DNA to produce *messenger RNA* (*mRNA*), a *ribonucleic acid* (*RNA*) molecule. Second, the ribosome *translated* the mRNA into a chain of amino acids that folds into a 3-dimensional functional protein outside the nucleus.

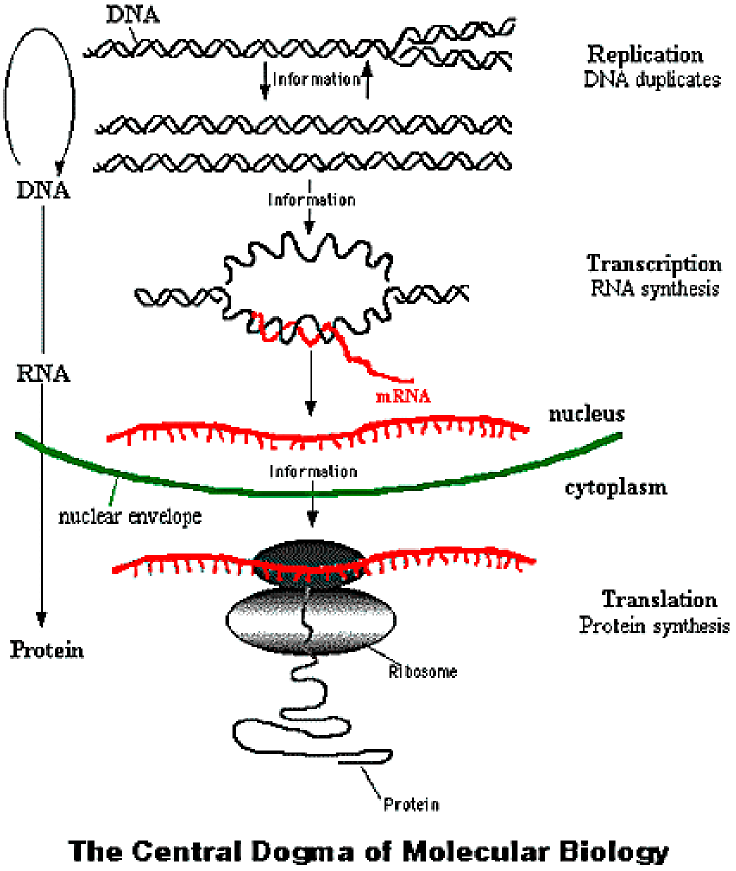


Figure 3.1: The central dogma of molecular biology.

Figure from http://www.accessexcellence.org/RC/VL/GG/central.html

Biological data are primarily organised around DNA and RNA (or mRNA) nucleotide sequences, genes, protein sequences, and 3D protein macromolecular structures; they account for 280 out of 500 (56%) biological databases registered at DBCAT catalogue [DBBG00]. The remaining 44% of the biological databases contain related literatures, mapping information of genes to the genomes, and other miscellaneous information.

3.1.2 Biological Database Systems

Biological data management systems usually take the form of publicly accessible biological databases [Ste03]. They include primary sequence databases, protein structure databases, gene expression databases, micro-array databases, databases of protein-protein interactions, and a large number of specialist databases. As of October 2005, the Molecular Biology

Database collection [Gal06] listed in total 858 databases classified into 14 categories and DBCAT listed 500 databases classified into 7 categories [DBBG00].

Web-based integration systems, either in the form of integrated query systems or data warehouses provide virtual or materialized access to the major primary databases [DMM$^+$03]. Virtual integration systems, also known as federated databases, provide a software middle layer to query multiple primary databases and extract relevant data into consolidated reports. Examples of virtual integration systems include DiscoveryLink [HSK$^+$01], Kleisli [Wong01], SRS [ZLAE02], and Entrez [SEOK96].

Materialized integration approach adopts a persistent storage of the data using a data warehouse. Examples of biological data warehouses are a gene expression data warehouse [MT01], GIMS - a genomic data warehouse [CPW$^+$01], a microarray data warehouse [FHB$^+$02], Ligand data warehouse [FCM$^+$04], a general sequence information data warehouse [SHX$^+$05], a genome data warehouse [KKSL$^+$04]. Organised around specific subject, the goal of constructing a biological data warehouse is to facilitate integrative analysis, summarization of information, and extraction of new knowledge hidden in the data [BK04, KB05]. This in turn, depends on the presence of clean, up-to-date, and well-organised data and is particularly difficult in a warehouse environment due to the diversity and distribution of the biological data from external sources. Therefore, data cleaning is an essential component in the biological data warehousing framework. For instance, Figure 3.2 shows that data cleaning is required in the data retrieval and update stages as well as annotation stage of BioWare – a biological data warehousing system [KKS$^+$04].

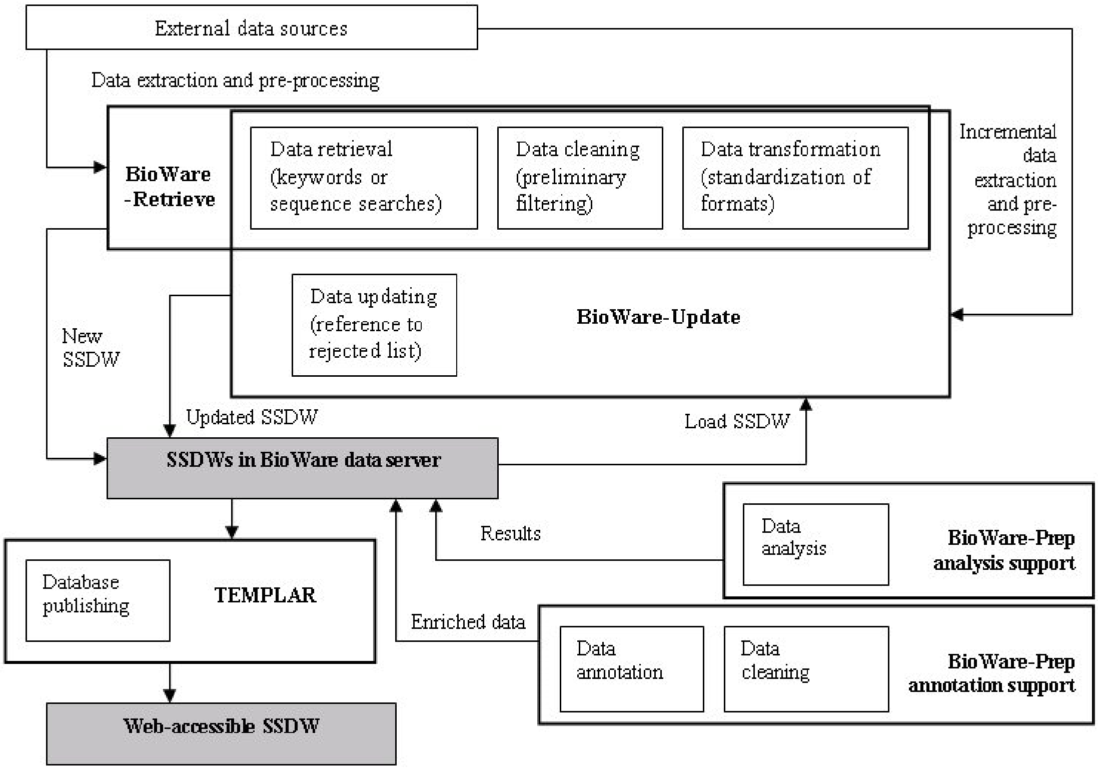

External data sources
Data extraction and pre-processing
BioWare -Retrieve
Data retrieval (keywords or sequence searches)
Data cleaning (preliminary filtering)
Data transformation (standardization of formats)
Incremental data extraction and pre-processing
Data updating (reference to rejected list)
BioWare-Update
New SSDW
Updated SSDW
Load SSDW
SSDWs in BioWare data server
Results
Data analysis
BioWare-Prep analysis support
Database publishing
TEMPLAR
Enriched data
Data annotation
Data cleaning
BioWare-Prep annotation support
Web-accessible SSDW

Figure 3.2: The data warehousing framework of BioWare

3.1.3 Sources of Biological Data Artifacts

Many reasons account for the presence of data artifacts in biological databases. Biological database records are primarily collected through direct submissions by the worldwide experimentalists and sequence centres, bulk submissions from high-throughput sequencing projects, or data exchanges between the databases. Adequate quality control of the submission process is often lacking, and therefore the correctness of the submitted data is not assured. Erroneous data may be mistakenly submitted, especially in projects that produce voluminous data.

Different molecular databases have different data formats and schemas, and nomenclature is not standardized across databases. This introduces high level of information redundancy because the same sequence may have inconsistent, overlapping, or partial information in heterogeneous representations that cannot be easily merged. Some of the major databases update one another, replicating partial or full entries from one database to another. For example, GenBank [BKL+06] contains data from direct submissions and bulk daily updates from DDBJ [OSGT06] and EMBL [KAA+05], and vice versa. Replication of data also happens due to the annotation of same sequences by different groups, submission of the same sequence to different databases, or even re-submission of the same sequence to the same database either by same or different authors.

In addition, the primary sequence records in the databases are often enriched with additional functional and structural information through manual annotations. The Swiss-Prot section of the UniProt database (UniProtKB/Swiss-Prot) is hand-curated by expert annotators from the Swiss Institute of Bioinformatics in Switzerland [WAB+06], while GenBank and EMBL allow sequence submitters to modify their sequence records with additional information. Random errors may escape the inbuilt quality control mechanism of human annotation and submitting authors not familiar with data models may input correct information in the wrong record fields. Also, sequence annotations may not be consistent across databases and interpretations of the same biological entity may differ. Consequently, different records describing the same sequences sometimes provide discrepant information.

Numerous biological databases consist of derived data generated from biological analysis, data mining or computational annotations. One example is the TrEMBL section of the UniProt database (UniProtKB/TrEMBL); the functional annotations of new protein sequences are computationally inferred from similar protein sequences. Computational annotations are not completely accurate and are subject to certain degree of annotation errors. For example, Wieser *et al.* [WKA04] detected mis-annotations in UniProtKB/TrEMBL by cross-validating its predictive models with Uni-ProtKB/Swiss-Prot.

3.2 Motivation

Multiple sources of introducing artifacts to biological database systems, combined with the lack of adequate quality control cause a manifestation of low quality data. Data artifacts typically affect more than 10% of the records in a biological data set. An earlier study of swine lymphocyte antigens (SLA) showed that of all records extracted from public databases, 17% of records contained at least one type of artifacts [SKB00]. Korning *et al.* [KHRB96] extracted data from GenBank for the analysis of *A. thaliana* splice sites. They reported that more than 15% of the *A. thaliana* records from GenBank must be removed in order to achieve a reasonable prediction of the splice sites.

The presence of data artifacts in public sequence data has been known for a long time and individual artifacts have been reported, but a complete analysis of the data quality issues pertaining to biological data is lacking. Overton and Haas [OH98] studied sequence structure violations in molecular databases. The presence of annotation errors was discussed [Bre99, BC01, GAD$^+$02, ITA$^+$03, MNF03]. Studies of contaminated sequences were conducted [LKSV92, SFM$^+$99, GAA$^+$00]. Some studies have focused on the analysis of specialized datasets. Expressed sequence tags (EST) were analysed for presence of contaminations [SS03], and the extraction of high quality datasets [Tha99, PHBR04].

Designing data cleaning remedies requires investigation of the sources of artifacts, the mechanism of their introduction into the databases, and their manifestation in the databases. Understanding these factors also provides an insight into possible limitations of data cleaning methods.

3.3 Classification

In total, we observe 11 types and 28 subtypes of artifacts across major nucleotide and protein sequence databases. Classified according to their physical and conceptual sources, they represent the "roadmap" for cleaning biological data.

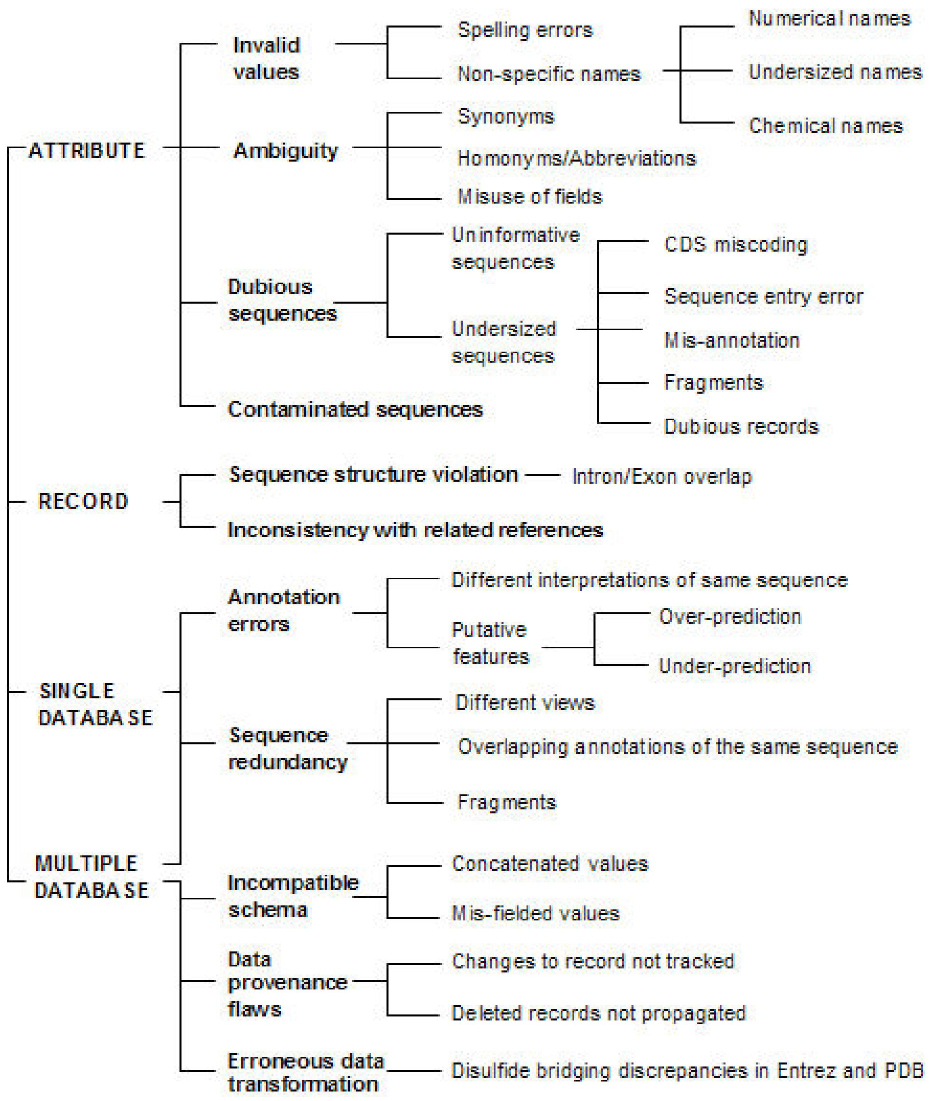


Figure 3.3: The 4 levels physical classification of data artifacts in sequence databases

The first classification grouped data artifacts according to their presence in data items at four levels of detail – individual attributes, individual records, individual databases, and multiple databases (Figure 3.3). Such separation follows the intuitive progression of the data cleaning process for resolving artifacts at data entities of varying granularity. For example, it makes sense to correct individual attribute values before cross-referencing two or more attributes for discrepant information. The classification also reflects the complexity of the data cleaning steps. Unlike duplicate detection in a single database, the same operation for multiple databases of different schema requires additional mapping steps to identify the corresponding attributes.

The second classification differentiates artifacts by their conceptual sources, distinguishing artifacts originating from annotation or implementation processes with those inherently parts of the imprecision of biological knowledge (Figure 3.4). This classification groups the artifacts according to their conceptual sources of bioinformatics and biological origins. Artifacts of bioinformatics sources are imperfections of the information systems that deal with how sequences are collected, annotated, and stored in databases. These are data management issues from the viewpoint of bioinformatics. Related artifacts are either syntactic (related to the design and structure of the databases) or semantic (related to the meaning or interpretation of the sequences). Biological artifacts are mistakes due to the shortcomings of biology as an empirical science. They reflect the complexity of biological systems.

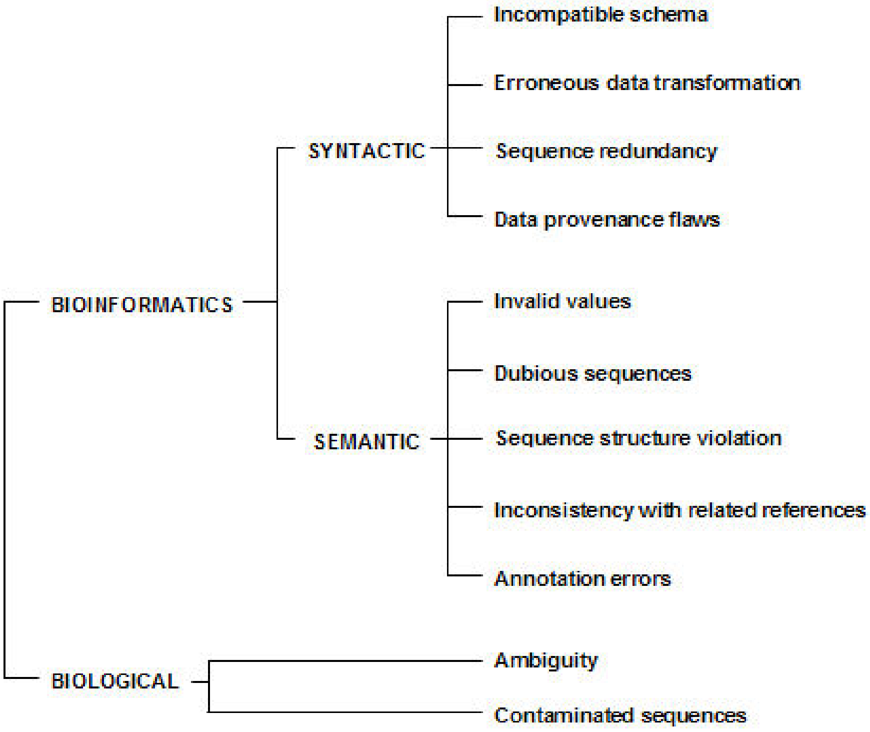


Figure 3.4: The conceptual classification of data artifacts in sequence databases

The next few sections detail the 11 types and 28 subtypes of artifacts identified from our study. Examples given in this study are extracted using a number of heuristic methods and programs which we developed based on domain knowledge. These examples are found in the NCBI Entrez searchable nucleotide and protein databases [SEOK96] and the UniProt Knowledge Base (comprises of UniProtKB/Swiss-Prot and UniProtKB/TrEMBL). The nucleotide databases accessible via Entrez include GenBank, EMBL, DDBJ, RefSeqs (NCBI-curated non-redundant set of reference sequences) [PTM05], USPTO (sequences submitted to U.S. Patent and Trademark Office), and TPA (third party annotated sequences). The NCBI entrez searchable protein databases include GenPept (sequence data from the translated coding regions from DNA sequences in GenBank, EMBL, and DDBJ) as well as protein sequences submitted to Protein Information Resource (PIR) [WYH$^+$03], SWISS-PROT [BBA$^+$03], Protein Research Foundation (PRF), and sequences from solved structures of Protein Data Bank (PDB) [DAB$^+$05].

Most biological database records discussed in this section are collected into the online BioDArt catalogue (http://antigen.i2r.a-star.edu.sg/BioDArt).

3.3.1 Attribute-level artifacts

Attribute level artifacts are field values with uninformative, invalid, erroneous or ambiguous content. We observed four main types of attribute level artifacts - invalid attribute values, ambiguous attribute values, dubious sequences, and contaminated sequences.

3.3.1.1 Invalid values

Many attributes of a nucleotide or protein sequence record are free-texts, and therefore not restricted to any integrity, format or functional dependency constraint. These attributes are prone to data entry or typographical errors such as mis-spellings.

Spelling errors

Spelling errors are non-critical but affect the efficiency of keyword searches that demand exact matching of the input words. Simple spell-checking mechanisms distinguish spelling errors as entities which are missing in the spell-checkers' thesaurus. In biological texts, the continuous introduction of new names or identifiers for new genes, proteins, diseases and drugs, and the combinations of chemical names, pose difficulties in consolidating a complete thesaurus for spell checking. This affects the effectiveness of biomedical spell-checkers such as Spellex Bio-Tech (spellex.com/Products/biotech.htm), Free Medical Spell Checker (www.free-medical-spell-check.com), and Inductel Scientific and Technical Speller (www.inductel.com/spel_sci_spell.html); all 3 spell-checkers label "Cystin" - a cilia-associated protein described in 2002 [HMY$^+$02], as a misspelling for "Cysteine".

We detected 63 commonly occurring misspelled words through screening 35,322 *Human Lymphocyte Antigen* (HLA) nucleotide annotations using a general-purpose spell-checker, followed by manual verification. The verification step proved to be a crucial step; only 11% of the words identified are misspellings. Among those incorrectly detected by the spell-checker are new names, chemical names or linguistic irregularities such as "proliferations" and "leukaemias". Querying the Entrez nucleotide and protein databases with these 63 misspelled words returns 7,075 databases records (as of August 2006, data shown in BioDArt). For example, the word "mitochondrial" is commonly misspelled as "mitchondrial" or "mitochrondrial"; more than 600 records containing either of the two misspelled words.

Non-specific names

The ability to identify sequences through their names depends on the specificity of the gene or protein names assigned. Ideally, names should be uniquely associated with groups of related sequences. However, standardization of naming conventions for biological entities, such as in enzyme nomenclature (EC terms) is not completely established. Assignments of names to a gene or a protein are often left to the discretion of submitting authors who may give improper names to the sequences. Some protein names are numerical; they correspond to the locus of the coding genes or an arbitrary form of numbering decided by the experimentalists, others are undersized names comprising one or two letters, and some chemical rather than common names. For example, the Actin protein recorded in ACTM_HELER of UniProtKB/Swiss-Prot database is known as "M" which also denotes a gene accountable for the Newcastle disease virus (Locus 1312416B of PRF database).

Non-specific names can be detected by screening individual field names in databases for given formats (all digits or multiple "-"). Corrections require sequence databases to enforce format constraints on the values that are stored in these attributes. If implemented as quality control rules in the sequence submission tools such as that of GenBank *BankIT* or EMBL *Webin*, these constraints will prohibit submissions of attribute values beyond pre-determined limits.

3.3.1.2 Ambiguity

Incomplete standardization of naming conventions also results in a potentially wide spectrum of names describing the same sequence. Often enough, a sequence has multiple names (synonyms), and different sequences can share the same name (homonyms) or abbreviation. Synonyms and homonyms induce ambiguities in the identification of a particular sequence using keywords. Searching for a specific group of sequences using names with high homonymy usually result in a number of false matches.

Synonyms

Due to the lack of controlled vocabulary of biological entities, multiple names reference the same protein or nucleotide sequence. Excessive synonymy in namings causes information ambiguities. We screened 222,289 sequences recorded in the UniProtKB/SwissProt database (release 8.0) for the number of "Synonym" elements at each XML sequence record. UniProtKB/Swiss-Prot recorded 53% (122,099) proteins identifiable by up to 15 synonyms, among which 144 proteins have more than 10 synonyms (Figure 3.5, data in BioDArt). 81% of the synonyms are non-unique, meaning they are assigned to 2 or more proteins. For example, the "salivary acidic proline-rich phosphoprotein 1/2 precursor protein" (recorded in PRPC_HUMAN of UniProtKB/Swiss-Prot) is known as "PRP-1/PRP-2", "Pr1/Pr2", "Protein C", "PIF-S", "Parotid double-band protein", "Pa" as well as "Db-s".

In principle, this means that a large percentage of the protein sequences are identifiable by multiple names. The prevalence of multiple naming conventions for the same entities is not limited to this data set analysed. Rather, it is a common repercussion of incomplete standardized nomenclature and affects numerous sequence databases.

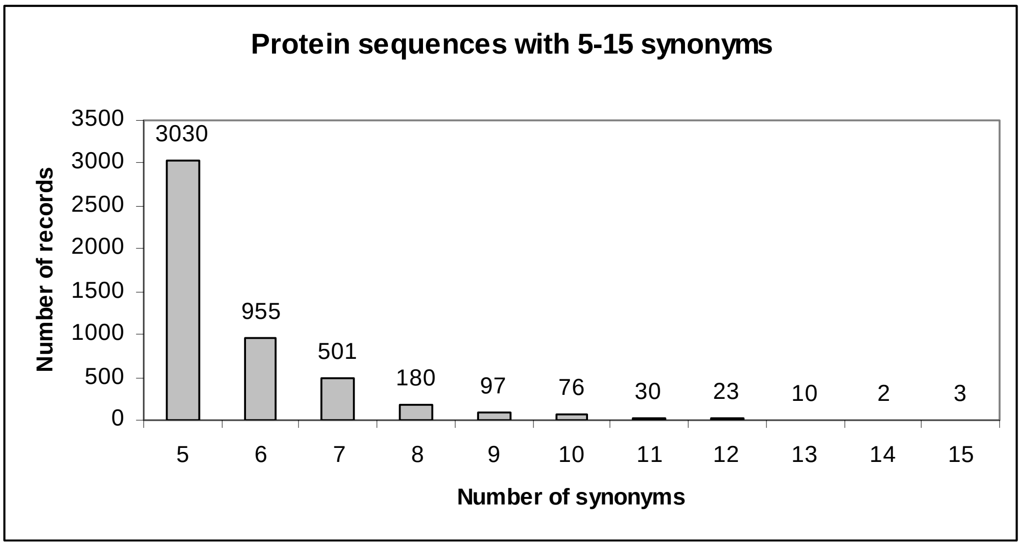


Figure 3.5: Protein sequences recorded at UniProtKB/Swiss-Prot containing 5 to 15 synonyms

Homonyms

The same names or abbreviations may refer to vastly different protein or nucleotide entities. While some homonyms are legitimate descriptors for sequences that are found in multiple organisms (such as the enzymes), others describe vastly different sequences. For example, 202 UniProt records containing the Gene field "GK" include glycerol kinases, glutamite kinases, guanylate kinases, a hexokinase, a Geko protein, Keratin, and a membrane spanning glycoprotein gK.

Again, we screened the UniProtKB/SwissProt database for the number of protein sequences bearing the same "Protein name". 72% of the proteins recorded in UniProtKB/Swiss-Prot have non-unique protein names. Among these, 14% of the protein names describe more than 100 other proteins (examples in BioDArt). For instance, 128 proteins recorded in UniProt is known as "Nucleoproteins"; significant number of these proteins is less than 5% similar in their sequences. The diversity of the proteins named after "Nucleoproteins" is not surprising, given that this general term refers to any protein associated with nucleic acids, and thus encompasses a variety.

Extensive synonymy and homonymy are consequences of incomplete standardization and lack of standard nomenclature which are critical guidelines governing the naming mechanisms of an empirical science [WMN01]. Agreeing on a particular gene or protein name facilitates research because it facilitates searches, as well as the mapping of sequences across different databases [NW03].

Research on the disambiguation of protein and gene names that aims at untangling the web of synonyms and homonyms [HDR01, YA03, PCG$^+$04] are directly applicable to resolving these ambiguities. Text-mining methods, particularly those devoted to solving the issue of named-entity recognition, provide dictionaries or thesaurus facilitating the clarification of gene and protein names [CHD05, ZZS$^+$04], although molecular terms typically mined from biomedical texts may have limited overlaps with databases [SC05]. In addition, isolated and usually species-dependent efforts can be noted (e.g., human gene names are reviewed by HGNC: http://www.gene.ucl.ac.uk/nomenclature/).

Misuse of Fields

Data entry mistakes of the sequence submitting authors to the wrong fields may produce ambiguous attribute values. For example, the protein name of UniProtKB/TrEMBL [BBA$^+$03] entry Q06524 was "NOTE THAT THERE ARE TWO OVERLAPPING ORFS ON THE OTHER strand" since 1996. The error was recently corrected (Release 44) to "Ypr151cp". We also reported a heat-shock protein sequence T45472 in PIR (version 79.1) with the gene name "Intron position not resolved (incomplete sequence)" which was later corrected.

Such erroneous attribute values take the form of attribute outliers in a database. However, in biological databases, the affected attributes are usually free-texts and therefore, the existing data profiling methods which rely on the numerical profiles of the attribute values to identify the attribute outliers are not directly applicable.

3.3.1.3 Dubious Sequences

Besides free text attributes, the protein or nucleotide sequence fields are also vulnerable to data entry problems and errors, with possible consequence of becoming useless for any form of data mining or analysis.

Uninformative Sequences

Part of the protein or nucleotide sequences contains abundant number of unknown residues “X” or unknown nucleotides “N”. A simple screening of all protein sequences recorded in UniProtKB/Swiss-Prot database identifies 13 proteins that have more than 30% unknown residues. For instance, the protein recorded in UN19_CLOPA contains the sequence “XXFESXEMR” which seems to be a motif representation rather than a protein sequence. A similar test on the dataset of 99 *Lymphocytic Choriomeningitis Virus* (LCMV) nucleotide sequences retrieved from Entrez identified 3 records [AH004715 AH004719 and AH004720] which are completely made up of “N” sequences. Three other nucleotide sequences [GI: 912860, 912868 and 912876] have 90% unknown sequence content.

These uninformative sequences are typically the result of erroneous sequence entry by the submitting authors. Such mistakes are easily avoidable if format constraints are adequately imposed upon new sequence submitted from the public to these databases.

Undersized sequences

Some protein or nucleotide sequences are too short for any meaningful form of analysis. We searched the Entrez system for sequences of a specified length. (e.g. query “4[SLEN]” where SLEN limits the search to sequence length). We observed 3,749 out of 10,350,551 proteins collected in the Entrez accessible protein databases are shorter than four residues (as of Aug 2006). In fact, 2,026 proteins contained only a single residue. Similarly, 1,957 out of 87,514,218 searchable nucleotide sequences from Entrez are shorter than six bases (as of Aug 2006). These undersized sequences are found in several major sequence databases (Figures 3.6 and 3.7, data in BioDArt).

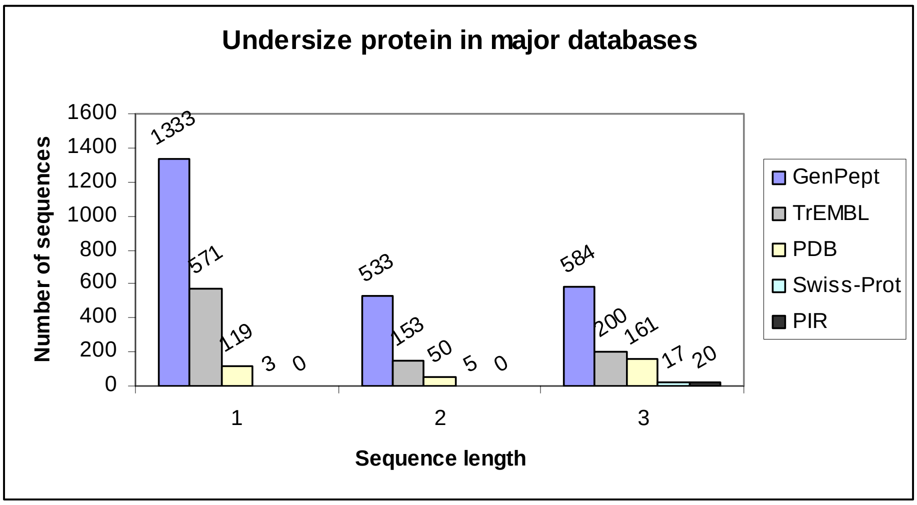


Figure 3.6: Undersized sequences in major protein databases

Some naturally occurring short sequences do exist such as thyrotropin-releasing hormone (TRH) tri-peptides and Glutathione. However, we found by manual inspection that with few exceptions, the undersized sequences in databases are erroneous. While some are erroneous submission of small sequence fragments, others are results of erroneous translation of the complementary nucleotide sequences to the proteins. For example, protein sequence which was produced by conceptually translating the “96 ..>98” region (where “..>” means the region extends beyond position 98) of its coding gene contain only a single amino acid. Also, there are cases of partial entry of sequences; submission of the full sequences are not completed.

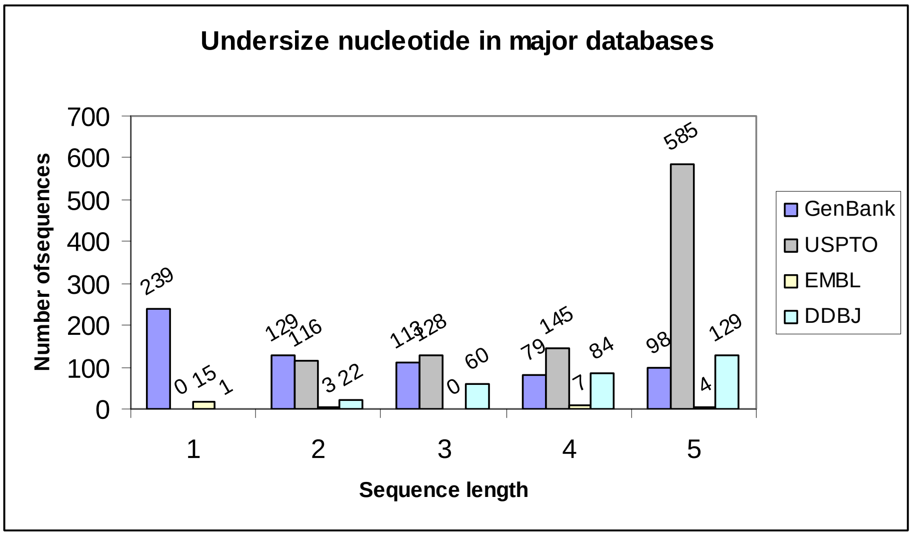


Figure 3.7: Undersized sequences in major nucleotide databases

An implementation to screen for undersized sequences is trivial. A more generalized approach, however, is to impose constraints on the minimum length of the sequences submitted to the primary databases.

3.3.1.4 Contaminated Sequences

Contaminated sequences are nucleotide sequences containing biological sequence segments of foreign origin. These foreign contaminants include insertions of transposable elements [Bin93], clones from heterologous sources [WDS+93], oligo-nucleotides such as adaptors, linker and PCR primers [CD04], bacteriophage [May78, LKP92, ŁCS+04], and cross-contaminated samples in the laboratories [DA95]; a comprehensive description of the various types of contamination is given in Sorek and Safer [SS03].

Sequence contamination studies have focused largely on contamination of cloning vectors. Vectors are agents that carry DNA fragments into a host cell. They are usually used for cloning (cloning vectors) or for expressing genes (expression vectors). The vector sequences probe and bind the DNA fragments at the 5' and 3' sites (Figure 3.8). The DNA fragment is then isolated from its vectors by cutting at the restriction enzyme sites. Some of the common vectors used in experiments include the plasmid, Lambda phage, cosmid and yeast artificial chromosome (YAC). In cases when the gene is not completely isolated from its vectors, the vectors become part of the gene submitted to the databases.

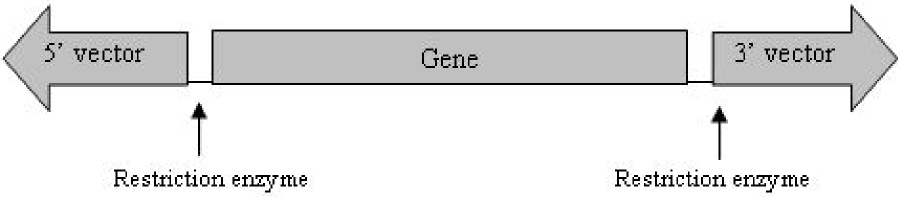


Figure 3.8: Nucleotide sequence with the flanking vectors at the 3' and 5' ends

The problem of vector contamination was discussed as early as in 1992 when 0.23% of 20,000 eukaryotic entries were found contaminated [LKSV92]. [SFM+99] reported that up to 0.36% of the sequences in GenBank were subject to similar contaminations. Vector contaminated sequences were identified by matching the sequences with a database of vector segments [MGB99]. This approach was adopted by NCBI VecScreen tool (www.ncbi.nlm.nih.gov/VecScreen) to screen all submitted sequences. However, our study indicated that VecScreen depends on the completeness of its UniVec cloning vectors database (ftp.ncbi.nih.gov/pub/UniVec).

Using BLASTN [AMS$^+$97], we matched 8,850 *C. albicans* nucleotide sequences retrieved from Entrez nucleotide databases with 18 cloning vectors commonly used for *C.albicans*. The program identifies 15 potential contaminations (data in BioDArt). For example, the regions at positions 1,737 to 1,825 - the 3' end of the *C. albicans* iro1 gene (Entrez GI: 9588659) is identical to a fragmented region of *C. albicans* vectors pDDB57 (Entrez GI: 6651387), and pGEM-URA3 (Entrez GI: 50363243). This indicates the iro1 gene submitted to EMBL is likely contaminated with part of its cloning vectors.

Though all new sequences submitted to the major databases are screened for vector contaminations, contaminations may still be missed because the vector databases are not updated. For instance, the cloning vectors of *C. albicans* were not found in the UniVec database, last updated in 1999.

3.3.2 Record-level artifacts

Conflicting information exists in the single record among two or more attributes – we call them the record-level artifacts. Two types of record-level artifacts are found in the sequence records – sequence structure violations and inconsistent content with related references.

3.3.2.1 Sequence Structure Violations

In eukaryotes, the typical structure of a gene follows order of the promoter region, the non-coding 5' untranslated region (5' UTR), alternating series of introns and exons, and ending with the 3' untranslated region (3' UTR) (Figure 3.9). The promoter, 3' UTR and 5' UTR regions contain regulatory elements that control protein synthesis. During splicing, the introns are removed from the primary transcript while the exons assemble to form the mature RNAs which are eventually synthesize to proteins.

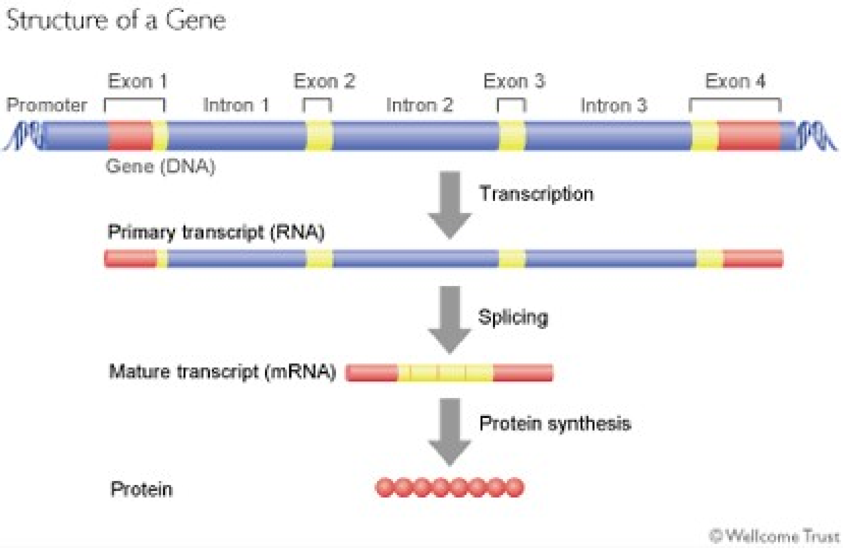


Figure 3.9: Structure of the eukaryotic gene containing the exons, introns, 5' untranslated region and 3' untranslated region

Figure from http://genome.wellcome.ac.uk/doc_WTD020755.html

Corresponding feature attributes of a gene database record contain non-overlapping positions ordered according to the gene structure. Incorrect specification of starting or ending positions of the regions, either by the database annotators or the submitting authors may cause erroneous overlapping regions, except for alternatively spliced cases.

The structural violations can be determined through comparing the positions and orderings of related features. The screening program takes into consideration domain rules for differentiating cases of alternative splicing from genuine intron/exon overlaps. Through this simplistic approach, we identify 9 fungal nucleotide sequences containing erroneous Intron/exon overlaps (Data in BioDArt). For example, an *Aspergillus niger* sarA gene submitted to GenBank (Entrez GI:1061033) has overlapping intron and exon 1, at regions 239..398 and 397..500 respectively.

Our method is limited by the accuracy and the completeness of the domain rules. In [OH98], these domain rules are machine-learned using case-based reasoning. The result is a rule-based sequence structure parser capable of identifying and correcting 60% of the complete coding sequences in GenBank.

3.3.2.2 Inconsistent with related references

Sequence annotations are typically derived from related literature references. Database curators enrich the sequence records with information that they have read from the reference articles. As with any manual annotation, the process is susceptible to human data entry errors, thus producing sequence annotations that are inconsistent with the complementary references provided.

In a study of Dengue virus, we observed mis-annotations in Swiss-Prot record P27915 and PIR record GNWVD3 [KHL$^+$06]. The NS1/NS2A and NS4A/NS4B junctions given in these Dengue type 3 complete RNA sequences did not match the regions given in the reference of these records [OS90]. While manual checking of such inconsistencies by cross-referencing the database content with their literatures is tedious, computational detection of discrepancies of the sequence annotations with its references is non-trivial and may require complex text-mining solutions. Nevertheless, research efforts such as the BioRAT information extraction system provide the basis for the development of a possible solution [CBLJ04].

3.3.3 Single Database level artifacts

Single database-level artifacts refer to redundancy and discrepancy of the information that exists in two or more records in the same database. Since these artifacts affect records from different databases, we also classify them as multiple-database level artifacts.

3.3.3.1 Annotation errors

Annotation is the process of assigning functional descriptions to new sequences. Both computational and manual annotation processes are susceptible to annotation errors – discrepant descriptors assigned to two or more sequences.

Different interpretation of the same sequence

Biologists studying the same sequences may provide different interpretations. A comparative study of the annotations by three different groups of 340 genes of *M. genitalium* genome showed that incompatible descriptions were assigned to 8% of these genes [Bre99]. In one example, the same mg085 protein was separately identified as a HMG-CoA reductase (EC 1.1.1.34), a NADH-ubiquinone oxidoreductase (EC 1.6.5.3), as well as an ATP/GTP enzyme. A study of the different annotations of the *C. trachomatis* genome also showed 37% of the dataset contains inconsistent functional assignments [ITA$^+$03].

[MNF03] estimated that annotation discrepancies affect 5% to 40% of the public protein and nucleotide sequences. The rate is a conservative under-estimation because in reality, the artifacts accumulate and propagate in multiple databases through transformations and data exchanges.

Putative Features

Due to the rapid accumulation of new uncharacterized sequences (only sequences; no functional or structural descriptions), there is a widespread use of computational annotation methods. Most of them rely on the inferences from homologous sequences. This depends on the assumption that highly similar sequences have the same biochemical properties and functions. Nevertheless, whether two sequences are "homologous" depends on individual judgements, thus giving rise to annotation discrepancies.

Putative features refer to attributes of the query sequence that are inferred from attributes of similar sequences found in the database. The correctness of putative features largely depends on the thresholds used as similarity cut-offs. In some cases, even the highest matching sequence from database search may have weak sequence similarities and therefore might not share similar functions with the query sequence. Comprehensive overviews of the limited accuracy of putative features are given in [Bor01, GAA$^+$00].

The result of computational inferences can be an under- or over-prediction. Under-prediction means only one or a few of the properties of the matching record are adopted to the query sequence. An over-predicted annotation assignment transfers all annotations of the best matching sequence without additional verification. Some active sites dependent functional properties cannot be extrapolated from similar sequences and "blind" inference can cause erroneous functional assignment.

Also, it is not always obvious to the database users which attributes are putative. The users are therefore unable to judge the credibility of a given record. Indicating the putative features of a submitted sequence to public domain databases is encouraged yet not strictly enforced. GenBank depends entirely on the goodwill of sequence submitters to label the putative features. In curated databases such as UniProtKB/Swiss-Prot, descriptions of protein sequences whose functions are putatively inferred start with the word "Putative". Putative features are also separately marked either with "Potential", "By Similarity" or "Putative" (according to section 2.4 of www.expasy.org/sprot/userman.html#contents).

While annotation errors and discrepancies affect a significant percentage of the publicly available protein sequences, data cleaning methods that address them are lacking. Among the very few known strategies are [WKA04, KL05]. Aggravating the data quality problem, annotation errors are not isolated cases of the affected sequence but proliferate to new sequences with functional assignments drawn from the mis-annotated sequences. Details of the proliferation rate of mis-annotations are discussed in Giks *et al.* [GAD+02].

In Chapter 4 and 5, we introduce a general attribute outlier detection method that is capable of accurate detection of annotation discrepancies in protein annotations. Unlike classical outlier detection methods, we leverage on the correlation between attributes to determine outlier entities.

3.3.3.2 Sequence Redundancy

As discussed in section 3.1.3, various factors contribute to sequence redundancy. In order to estimate the extent of redundancy in major sequence databases, we screen 4,431 fungal sequences in Entrez nucleotide databases using a simple rule:

Two records are duplicates if the sequences are identical and they belong to the same species.

The same rule is used in the UniParc database to merge redundant sequences from PIR and Swiss-Prot [LDB+04]. Of the records from 4 nucleotide database sources - EMBL, DDBJ, GenBank, and RefSeq, 7% contained sequences described in other records. If duplicate sequences identified from this naïve rule are trivially merged, the redundancy of the dataset will reduce by 81% (from 299 records to 58 records). However, manual verifications show that some sequences identified by this naïve rule sometimes belong to different strains, and therefore may have different feature descriptions. This difference, however, is not usually obvious from inspecting the content of the database records. It is also not obvious if identical sequences of dissimilar isolates, clones or specimens come from the same strain.

In general, whether to merge the duplicates detected depend on factors beyond on the similarity of the records. Rather, different types of duplication exist. Similarly, in protein, we observed four different types of duplication (details in Chapter 6). As such, existing duplicate detection methods that consider duplication as a Boolean property are not directly applicable.

3.3.4 Multiple Database level artifacts

Massive transformations occur during exchange and integration of sequence records from multiple databases. Defective data transformation from syntactically different databases may result in systematic errors.

3.3.4.1 Incompatible schema

Sequence data are repeatedly copied, corrected, and transformed among heterogeneous databases. Matching the fields in databases with dissimilar schemas sometimes introduce artifacts because of schema mismatches.

Concatenated values

During data transformation, 2 or more attributes are sometimes concatenated into a single attribute when the target database schema does not contain structures for some of the fields in the original schema. For example, the function, sub-cellular location, tissue specificity, and other functional descriptors of a UniProtKB/Swiss-Prot sequence map to the Seqdesc_comment attribute of the corresponding Entrez record (Figure 3.10).

```
<comment type="function">
   <text>Shows anti-epileptic activity. Shares high homology
with depressant insect toxins, but shows very weak toxicity
against mammals an insects and no obvious symptoms on
insect larvae</text>
</comment>
<comment type="subcellular location">
   <text>Secreted protein</text>
</comment>
<comment type="tissue specificity">
   <text>Expressed by the venom gland</text>
</comment>
<comment type="mass spectrometry" mass="6730.4"
method="Electrospray">
   <location>
     <begin position="22"/>
     <end position="82"/>
   </location>
   <note>Ref.1</note>
</comment>
<comment type="similarity">
   <text>Belongs to the long (4 C-C) scorpion toxin
superfamily. Sodium channel inhibitor family. Beta
subfamily</text>
</comment>
```

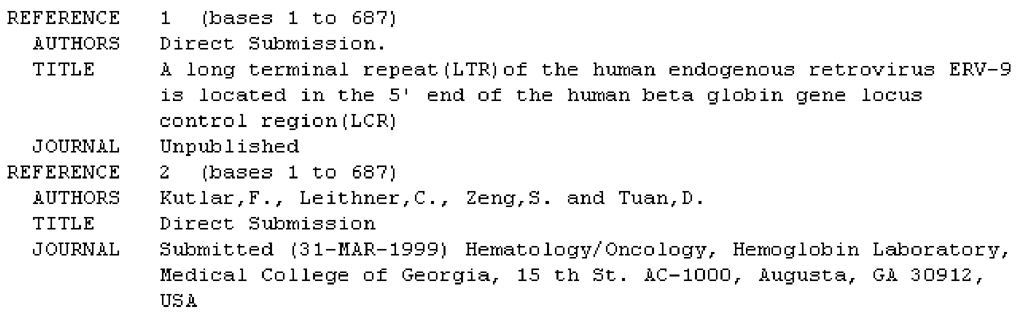
```
<Seqdesc>
   <Seqdesc_comment>[FUNCTION] Shows anti-epileptic
activity. Shares high homology with depressant insect toxins,
but shows very weak toxicity against mammals an insects
and
no obvious symptoms on insect larvae.
   </Seqdesc_comment>
</Seqdesc>
   <Seqdesc_comment>[SUBCELLULAR LOCATION]
Secreted protein.
   </Seqdesc_comment>
</Seqdesc>
<Seqdesc>
   <Seqdesc_comment>[TISSUE SPECIFICITY] Expressed
by the venom gland.
   </Seqdesc_comment>
</Seqdesc>
<Seqdesc>
   <Seqdesc_comment>[MASS SPECTROMETRY]
MW=6730.4; METHOD=Electrospray; RANGE=22-82;
NOTE=Ref.1.
   </Seqdesc_comment>
</Seqdesc>
<Seqdesc>
   <Seqdesc_comment>[SIMILARITY] Belongs to the
long (4 C-C) scorpion toxin superfamily. Sodium channel
inhibitor family. Beta subfamily.
   </Seqdesc_comment>
</Seqdesc>
```

Figure 3.10: The functional descriptors of a UniProtKB/Swiss-Prot sequence map to the comment attributes in Entrez

Such compounded fields reduce the structure of a database and cause difficulties in extracting specific data. In [HKK03], we extracted gene names and protein synonyms from Uni-ProtKB/Swiss-Prot and LocusLink [PM01] records for purpose of text-mining protein-protein interactions from PubMed abstracts. This extraction was complicated because multiple names were concatenated in a single field using "and", "or" or commas as separators. In Uni-ProtKB/Swiss-Prot, this problem was resolved in later releases.

Mis-fielded values

If the target database schema does not contain attributes designed for a particular piece of information, such information are inserted into the wrong fields. For example, the data schema of Entrez records does not contain separate fields for the sources of sequences from direct submission. The date of submission and the author address are therefore inserted into the reference TITLE and JOURNAL attributes (Figure 3.11).



```
REFERENCE   1  (bases 1 to 687)
  AUTHORS   Direct Submission.
  TITLE     A long terminal repeat(LTR)of the human endogenous retrovirus ERV-9
            is located in the 5' end of the human beta globin gene locus
            control region(LCR)
  JOURNAL   Unpublished
REFERENCE   2  (bases 1 to 687)
  AUTHORS   Kutlar,F., Leithner,C., Zeng,S. and Tuan,D.
  TITLE     Direct Submission
  JOURNAL   Submitted (31-MAR-1999) Hematology/Oncology, Hemoglobin Laboratory,
            Medical College of Georgia, 15 th St. AC-1000, Augusta, GA 30912,
            USA
```

Figure 3.11: Mis-fielded reference values in a GenBank record

Incorrect fitting of field values during data transformations reduced the effectiveness of obtaining specific and meaningful query results. When constructing a relational genomic data warehouse of database records extracted using Entrez, we had to differentiate between valid reference field values and those describing the sequence submission sources.

3.3.4.2 Data Provenance Flaws

Data provenance is the understanding of data origin and its transformation over a series of updates. Data provenance information is important for data miners who need to keep track of critical changes to the data records which may affect the analysis results derived from the older versions.

In some sequence databases (PDB, PIR, among others), data provenance information is not explicitly captured. Minimal revision history such as dates of modification and indications of changes in main sections (annotation or sequence) were provided. Database users cannot reconstruct how a sequence was added to the database and the subsequent changes (corrections, additions, and deletions) made to a given sequence record. Comparing the current release of a data record with its previous version in earlier releases usually provides the details of changes. Yet, the archives of past releases of these databases are not necessarily available.

However, recent efforts by UniProt significantly improved the data provenance concern in the bioinformatics data community. The UniSave annotation version database accurately documents the modifications that have been made to every UniProt proteins [LNZA06].

As there was no agreement among different database groups to maintain the integrity of data across multiple databases, modifications and deletions made to a record in one database are also not necessarily propagated to other databases.

3.3.4.3 Erroneous Data Transformation

Table 3.1: The disulfide bridges in PDB records 1VNA, 1B3C and corresponding Entrez record GI 494705 and GI 4139618

Positions of disulfide bonds in PDB record IVNA	Positions of disulfide bonds in Entrez record 494705
(12, 65)	(16, 41)
(16, 41)	(25, 46)
(25, 46)	(29, 48)
(29, 48)	
Positions of disulfide bonds in PDB record IB3C	**Positions of disulfide bonds in Entrez record 4139618**
(11, 64)	(15, 40)
(15, 40)	(24, 45)
(24, 45)	(28, 47)
(28, 47)	

Errors may be generated from system defects in data integration mechanisms. The transformation in Entrez caused records imported from PDB to show incorrect disulfide connectivity patterns in the features section of imported records. Table 3.1 shows an example of missing disulfide bridges in records imported from PDB records. In all these cases, one end of the missing bridge occurs on the last residue of the sequences.

3.4 Applying Existing Data Cleaning Methods

We summarize the possible data cleaning remedies for each of the 11 types of artifacts and their limitations in Table 3.2. Some artifacts can be detected using simple format constraints of individual fields (*Undersized sequences*, *Uninformative sequences*, *Non-specific names*), while some require serious data processing in order to be identified (*Contaminated sequences*, *Sequence structure violations*) and others require certain level of expert verifications (*Spelling errors*, *Synonyms*, *Homonyms/Abbreviations*). Artifacts such as *Incompatible schema*, *Data transformation errors*, and *Data provenance flaws* are most appropriately resolved by the database developers through re-design of the systems and databases. Although advancement in text-mining may facilitate cross-checking, at this point of time, *inconsistency with related references* are still determined primarily through manual checks.

Table 3.2: Summary of possible biological data cleaning remedies

Types of artifacts	**Possible remedies**	**Limitations or requirements**
Spelling errors	Spell checkers	New names and chemical names are often mistaken as misspellings.
Non-specific names	Format constraints	-
Ambiguity	Disambiguition of gene and protein names [HDR01, YA03, PCG^{+}04]	Detection is possible but correction of names may require expert knowledge.
Dubious sequences	Format constraints	-
Vector contaminated sequences	Sequence similarity searches against updated vector database [VecScreen]	Accuracy depends on the currency of vector databases
Sequence structure violations	Detection using domain rules learned from known structural violations [OH98]	Alternative splicing cases can be mistaken as violations.
Inconsistency with related references	Cross-checking of database records with full reference texts	Mainly manual.
Annotation errors	**Outlier detection**	**Existing class outlier detection methods are not directly applicable**
Sequence redundancy	**Duplicate detection**	**Multiple types of duplicates exists**
Incompatible schema	Re-design of schema	-
Data provenance flaws	Database maintenance using metadata.	-
Erroneous data transformation	Correct system bugs	-

This thesis focuses on the *Annotation errors* and *Sequence Redundancy*. If an annotated attribute exhibits abnormal correlations with other annotated descriptors of the same sequence, it is likely that the outlier attribute is erroneous. Based on this idea, outlier detection approaches clearly provides a way to determine annotation errors. However, existing outlier detection methods target at class outliers and do not regard the correlations between the attributes. Similarly, existing duplicate detection methods do not take into account the presence of multiple types of duplicates. In the next 3 chapters, we introduce new correlation-based data cleaning methods which enable the detection of these 2 types of artifacts. These methods, however, are general and are not restrictively applicable to biological data.

3.5 Concluding Section

As biological data continue to accumulate exponentially, biological sequence databases are confronted with a critical need to tackle the corresponding depreciation of data quality. Our analysis of the presence of data artifacts indicates that the data quality problem is a collective result of 11 types and 28 sub-types of artifacts at the field, record, single and multiple-database levels. It is also a combinatory problem of the bioinformatics that deals with the syntax and semantics of data collection, annotation, and storage, as well as the complexity of biological data.

An assembly of data cleaning methods is required to eliminate these artifacts. Some artifacts can be resolved partially or completely using existing data cleaning methods; certain heuristics methods which we have developed to screen the presence of artifacts in biological data sets and databases can be expanded into complete data cleaning solutions. For more complicated artifacts such as annotation errors and sequence redundancy, correlation-based data cleaning approaches explore in the remaining chapters of this thesis provides new strategies for detecting the outliers and duplicates.

Chapter 4: Correlation-based Detection of Attribute Outliers using ODDS

A journey of a thousand miles begins with a single step.

Confucius
Teacher and Philosopher (BC 479-551)

An outlier is an object exhibiting alternative behaviour in a data set. It is a data point that does not conform to the general patterns characterizing the data set. Outlier detection has important applications in data cleaning. It provides the basis of finding errors and discrepancies that account for data noises reducing the accuracy of the mining and analysing large volume of data.

Existing outlier detection methods focus on class outliers and research on attribute outliers is limited, despite the equal role attribute outliers play in data quality depreciation. Methods for detection of class outliers rely on rarity observed from the distribution, density or distance of the tuples to determine outlier-ness. For attributes, however, rarity does not necessarily equate abnormality. Rather, an attribute outlier is defined by its deviating correlation behaviour with respect to other attributes.

In this chapter, we propose a novel correlation method to detect attribute outliers in databases. The method, which we call ODDS (for Outlier Detection from Data Subspaces) regards attribute outlier as a local deviator and systematically searches the data subspaces to identify possible outliers. ODDS is a general method, therefore both non-biological and biological applications are discussed.

We formulate three metrics to evaluate outlier-ness of attributes, and introduce an adaptive factor to distinguish outliers from non-outliers. We also devise two filtering strategies to reduce the complexity of the ODDS algorithm. Experiments with a synthetic data set indicate that our proposed ODDS method achieves an accuracy of up to 88%. Following the discussion in Section 3.4, we demonstrate that ODDS is effective in identifying annotation errors in the UniProt protein database.

4.1 Introduction

Outlier detection is the process of identifying outliers that do not conform to the general behaviour of the data set. Contrary to other data mining methods aim at determining predominant patterns in majority of the data, outlier detection methods target the under-represented, abnormal patterns that deviate from the rest of the data. These patterns are often the consequences of data artifacts such as errors, discrepancies, spelling errors, illegal values, and multiple types of violations. Therefore, outlier detection is the basis for data noise reduction in data cleaning, and is an imperative step towards improving the accuracy of any data mining and analysis applications.

Besides data cleaning, outlier detection is also the foundation of applications such as fraud detection, stock market analysis, intrusion detection, network sensors analysis, and email spam detection where the presence of outliers suggest abnormal signals entailing special notice and further investigation [Esk02, LSM99, PPKG03].

4.1.1 Attribute Outliers and Class Outliers

Existing outlier detection methods focus primarily on *class outliers* - multivariate data points (tuples) which do not fit into any class by definitions of distance, density, or nearest neighbour [ZW04]. Comparatively, data cleaning research have overlooked developments in the detection of *attribute outliers* - univariate points that exhibits deviating correlation behaviour with respect to other attributes although for a number of reasons, detecting attribute outliers is an equivalently important problem in data cleaning.

First, class outliers are often the result of one or more attribute outliers. Correcting or eliminating only the affecting attributes rather than the tuples has the advantage of fixing the abnormal behaviours while retaining information. Second, even when attribute outliers do not affect class memberships, they may still interfere with the data analysis mechanisms as data noise. Third, for many real-world data sets that do not contain class attributes, it is still meaningful to identify attribute outliers because they are sources of errors. One example is the UniProt database which contains the functional, structural, and physico-chemical descriptions of proteins [WAB$^+$06]. Though there is no meaningful class attribute for proteins, maintaining correctness of every detail provided in these records is critical, given that the worldwide biological researchers reference them extensively for analysis and experimental planning.

The nature of the outlier patterns associated with class and attribute entities differ. A common approach for detecting class outliers is to cluster the tuples in order to isolate the class outliers as singletons. However, applying the same clustering technique to attributes is only effective in finding rare attribute values, which are not necessarily attribute outliers. While we detail this observation in Section 4.2.2, it suffices to mention here that for attribute outliers, rarity does not equate abnormality. Rather, the deviating correlation behaviour of an attribute with respect to other attributes within a localized subspace defines its outlier-ness.

4.1.2 Contribution

This chapter focused on a novel correlated-based detection method for attribute outliers. We call the method **ODDS** to denote attribute **O**utlier **D**etection from **D**ata **S**ubspaces. In contrast to a global property that is applicable to all dimensions of the data set, we consider an attribute outlier as a localized deviator of a particular subspace. Our notion of attribute outlier-ness is therefore a bivariate property of both the *target attribute* and a *correlated neighbourhood*. The ODDS algorithm effectively iterates through the data subspaces to compute the outlier scores for each bivariate tuple of attribute outliers. The ODDS method is effectively applicable to the identification of annotation errors in a large protein database. Specific contributions of this work include:

1. A formal definition of attribute outliers based on the correlation behaviour of attributes in data subspaces.
2. Three new metrics *O-measure*, *Q-measure* and *O_f-measure* quantify the deviating correlation behaviour of an attribute. O-measure is the most accurate while Q-measure is computationally less intensive. O_f-measure is devised for sparse data sets containing vast occurrences of rare attribute values which are not outliers.
3. An adaptive *Rate-of-change* factor selects optimal thresholds for distinguishing the outliers from non-outliers in any given data set. These automatic and data-dictated thresholds remove dependency on user-defined parameter.

4. The ODDS algorithm systematically detects attribute outliers in data subspaces, together with two strategies to filter subspaces that do not contain attribute outliers.

The organization of the rest of this chapter is as follows. A motivating example is given in the next section. Section 4.3 details the formal definitions. In Section 4.4, we present the ODDS algorithm, and experimental evaluations follow in Section 4.5. We conclude in Section 4.6.

4.2 Background

Attribute outliers are generally under-studied in outlier detection research. Comparatively, methods for detecting class outliers are common; the data cleaning survey in Chapter 2 elaborates some of these methods. Among the few attribute outlier detection methods are distribution-based (section 2.3.2.1) and data polishing (section 2.3.1.2) approaches. In Section 4.5, we compare the performance of ODDS to the data polishing method using C4.5 as the classifier – the same strategy used in [ZW04].

4.2.1 Motivating Example

Table 4.1: World Clock data set containing 4 attribute outliers

	Country	State	City	Day	Time†	Weather
1	U.S.A	California	LA	Tue	8:40pm	Sunny
2	U.S.A	California	LA	Tue	8:40pm	Rainy
3	U.S.A	California	**Vancouver**Y	**Wed**Z	8:40pm	Sunny
4	U.S.A	California	LA	Tue	8:40pm	Storm
5	U.S.A	California	LA	Tue	8:40pm	Snow
6	Canada	British Columbia	Vancouver	Tue	8:40pm	Storm
7	Canada	British Columbia	Vancouver	Tue	8:40pm	Sunny
8	Canada	**California**W	Vancouver	Tue	8:40pm	Rainy
9	Canada	**B. C.**X	Vancouver	Tue	8:40pm	Rainy
10	Canada	British Columbia	Vancouver	Tue	8:40pm	Rainy
11	Micronesia	Ponape	Palikir	Wed	2:40pm	Storm

† Class attribute W, X, Y, Z Attribute outliers

We illustrate the rationale of key concepts in ODDS using a simple example in Table 4.1. The example is a World Clock data set, with the class attribute Time. It contains 4 attribute outliers in record 3, 8, and 9. 'B.C.' (*X*) is an unlikely abbreviation of British Columbia and

‘California’ in tuple 8 (*W*) is an erroneous data entry. While it is easy to segregate *X* from the distribution model of all attributes in the State dimension, discriminating *W* is more complicated because it seems to be a perfectly legitimate value in the same distribution. Similarly, the erroneous ‘Vancouver’ (*Y*) and ‘Wed’ (*Z*) are non-obvious.

Nevertheless, observations drawn from the supports or frequencies of attribute combinations provide hints for determining all these attribute outliers (Figure 4.1):

Observation 1: For attribute outliers, rarity does not equate outlier-ness. Tuples with one or more rare values may possibly be class outliers, but rare attributes are not necessarily attribute outliers. Consider Case C – although the tuple is a perfectly legitimate class outlier belonging to the *rare class* of ‘2:40pm’ in Table 4.1, individual attributes of the tuple - ‘Micronesia’, ‘Ponape’ and ‘Palikir’ are consistent in their correlation behaviour and are not erroneous, even if they are rare in individual dimensions of Country, State and City.

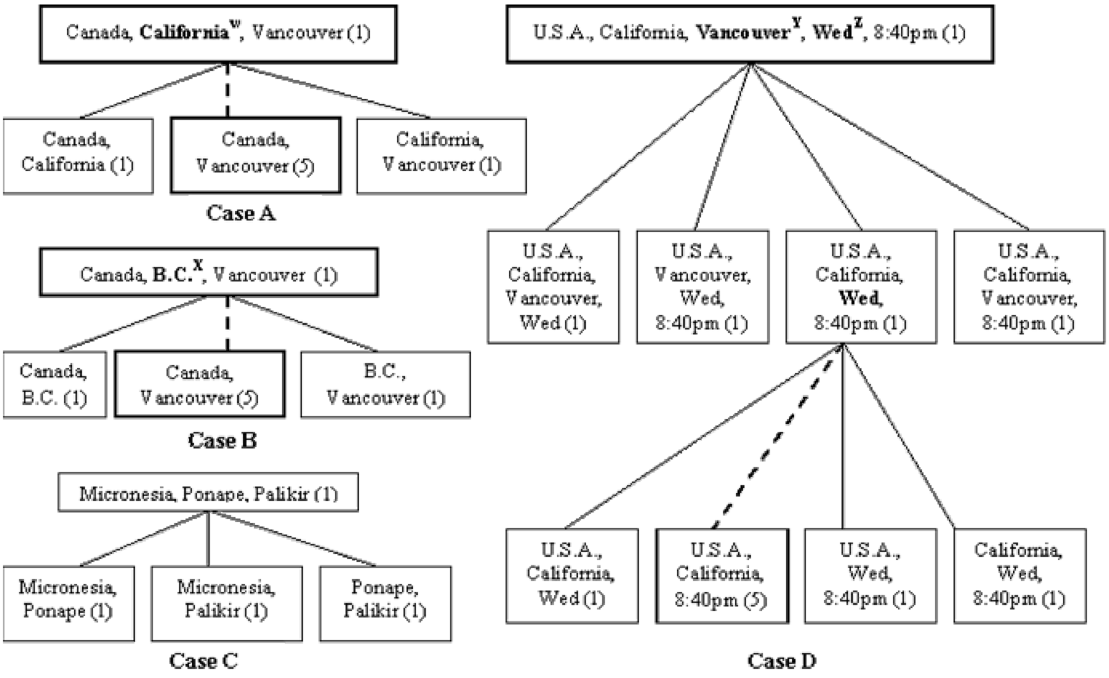

Canada, **California**^{W}, Vancouver (1)
Canada, California (1)
Canada, Vancouver (5)
California, Vancouver (1)
Case A
Canada, **B.C.**^{X}, Vancouver (1)
Canada, B.C. (1)
Canada, Vancouver (5)
B.C., Vancouver (1)
Case B
Micronesia, Ponape, Palikir (1)
Micronesia, Ponape (1)
Micronesia, Palikir (1)
Ponape, Palikir (1)
Case C
U.S.A., California, **Vancouver**^{Y}, **Wed**^{Z}, 8:40pm (1)
U.S.A., California, Vancouver, Wed (1)
U.S.A., Vancouver, Wed, 8:40pm (1)
U.S.A., California, **Wed**, 8:40pm (1)
U.S.A., California, Vancouver, 8:40pm (1)
U.S.A., California, Wed (1)
U.S.A., California, 8:40pm (5)
U.S.A., Wed, 8:40pm (1)
California, Wed, 8:40pm (1)
Case D

Figure 4.1: Selected attribute combinations of the World Clock dataset and their supports

In a similar biological example, 3 out of 208,005 tuples in the UniProt protein database (Release 7.1) contain the values <’Parkin’,‘PKRN’,‘S-nitrosylation’> for attributes

Protein name, Gene name and *Keyword* respectively. Despite rarity in their dimensions, they are not attribute outliers. In reality, few known protein sequences are associated with the Parkinson disease, but they are consistently known as Parkin, are products of PKRN gene, and are post-translationally modified by nitrosylation.

Observation 2:Attribute outliers are localized correlation deviators. Rarity may be a good indicator for class outlier-ness, but attribute outliers are determined by their deviating correlation behaviour within a localized subspace. In simple sense, an attribute outlier shows abnormal associations with other attributes within a data subspace. Consider Case A in Figure 4.1 –'California' is seldom associated with 'Canada' and 'Vancouver', compared to that of 'Vancouver' and 'Canada' which co-occur in 5 tuples. Comparatively, only 1 sub-tuple of <'Canada','California'> and of <'California','Vancouver'> exists. Intuitively, the greater the difference in the sub-tuple supports, the higher is the likelihood that 'California' is an attribute outlier in the combination <'Canada','California','Vancouver'>. This forms the basis of our outlier metrics. The same analogy identifies *X* in Case B.

Observation 3: A tuple may contain more than one attribute outliers. In real-world databases, a tuple often contain multiple attribute outliers. The multiplicity of attribute outliers within the same tuple interferes with the correlation patterns of one another, creating complications in segregating individual outliers. However, an attribute outlier can be isolated at lower dimensional attribute combinations. Consider Case D – the two attribute outliers separate when they are projected into different 4-attribute sub-tuples.

Besides other attribute outliers in the same tuple, noisy and uncorrelated dimensions may also interfere with the outlier detection algorithms. For example, the Weather dimension in Table 4.1 does not relate to any other attributes but contain non-deterministic/random values interfering with the correlation patterns. Since ODDS algorithm systematically explores individual subspaces for presence of attribute outlier, it naturally generates subspaces that do not contain any of the noisy dimensions.

To isolate the attribute outliers from non-outliers, users typically need to define a threshold. This is not viable in practice, given that the number of outliers in the real world dataset varies depending on the noise level of the data set and the data dimension under study. In the ODDS algorithm, the optimal threshold is determined from the maximal *Rate-of-change* which intuitively marks the point where sorted outlier scores drastically change. Rate-of-change is the natural boundary separating the outliers and non-outliers, and it removes the dependency of the outlier detection on any user-specified parameter.

Summarizing, unlike class outliers, rarity is not an adequate measurement of attribute outlier-ness. Rather, attribute outliers are local correlation deviators of a given subspace. Here, we formally define a correlation-based attribute outlier:

A correlation-based attribute outlier is an object which exhibits abnormal behaviour in a subspace of related objects.

An attribute outlier is a bivariate property of an attribute value and the subspace where it exhibits abnormal correlation. Therefore, attribute outlier detection methods involve finding both the attributes as well as the associated subspaces.

4.3 Definitions

In this section, we present all formal definitions used in ODDS.

4.3.1 Preliminaries

4.3.1.1 Support

Definition 1: Let R be a relation with m attributes. Let S be a projection of degree (v-u+1) on R over an arbitrary set of attributes A_u,..., A_v, $S = \pi_{A_u,\ldots,A_v}(R)$. The support of a tuple s in S, denoted by *sup*(s), is the count of the tuples in R that have the same values for attributes A_u,..., A_v as tuple s.

For example, consider the World-Clock relation R(Country, State, City, Day, Time, Weather) in Table 4.1, and a projected relation over three attributes, $S = \pi_{Country,State,City}(R)$. The support of tuple <'U.S.A', 'California', 'LA'> in S is 4 since tuples 1, 2, 4 and 5 in R have the same attribute values for Country, State and City. Similarly, sup(<'Canada', 'California', 'Vancouver'>) = 1.

4.3.1.2 Neighbourhood

Definition 2: Let tuple s=<a_u,…, a_v> be a tuple in the projected relation S. Without loss of generality, we consider A_v as the target attribute whose extent of deviation we are interested to determine. The neighbourhood of A_v w.r.t s is defined as *N*(A_v, s) = <a_u,…, a_{v-1}>. The support of *N*(A_v, s) is the count of tuples in R with the same values a_u,…, a_{v-1} for A_u,…, A_{v-1}.

Continuing from the last example, consider tuple s=<'Canada', 'California', 'Vancouver'> in the projected relation S. The neighbourhood of the State attribute in tuple s, denoted as *N*(State, s) is the sub-tuple <'Canada', 'Vancouver'>. Since the same values of 'Canada' and 'Vancouver' for attributes Country and City respectively are found in tuples 6, 7, 8, 9 and 10 of R, we have sup(*N*(State, s)) = 5.

4.3.2 Correlation-based Outlier Metrics

Our objective is to determine attributes which deviate from its neighbours in the projected relations. We formulate three metrics O-measure and Q-measure to quantify the extent of deviation.

4.3.2.1 O-measure

Definition 3: The O-measure (Outlier measure) of target attribute A_v w.r.t. s is defined as

$$O-measure\left(A_v,s\right)=\frac{\sum_{i=u}^{v-1}\sup\left(N\left(A_i,s\right)\right)}{\sup\left(N\left(A_v,s\right)\right)} \tag{4.1}$$

The lower the O-measure score, the more likely attribute A_v is an attribute outlier in s. Let us compute the O-measure of the attribute outlier *W* in Table 4.1. Let s=<'Canada', 'California', 'Vancouver'> be a tuple of $S=\pi_{Country,State,City}\left(R\right)$. The support of *N*(State, s) is

5 while sup(*N*(Country, s)) and sup(*N*(City, s)) are both 1. The O-measure of the State attribute w.r.t. s is (1+1)/5=0.4.

For comparison, we also compute the O-measure of the State attribute in tuple t=<'U.S.A', 'California', 'LA'>. We have O-measure(State, t) = (sup(*N*(Country, t)) + sup(*N*(City, t))) / sup(*N*(State, t)) = (4+5)/4 = 2.25. 'California' is an attribute outlier in attribute combination s but not in t, therefore O-measure(State, s) is relatively lower than O-measure(State, t). Recall that the outlier metric should not consider rare classes or events as attribute outliers. This is evident using O-measure where the high O-measure(Country, <'Micronesia', 'Ponape', 'Palikir'>) = 2 prevents the mis-interpretation of Micronesia as an attribute outlier.

4.3.2.2 Q-measure

The Q-measure of an attribute A w.r.t tuple s is defined as

$$Q - measure(A,s) = \frac{\sup(s)}{\sup(N(A,s))} \tag{4.2}$$

Let *a* be the attribute value of A. Q-measure is the *conditional probability* of a tuple having the value *a* for attribute A, given that the tuple has the same attribute values as the neighbourhood of A. Relating this back to the attribute outlier *W* in Table 4.1, Q-measure(State, <'Canada', 'California', 'Vancouver'>) = 1/5 = 0.2.

Computationally, it is less intensive to use Q-measure as the outlier detection metric because less calculation of the supports of neighbourhoods is required. This is however, at the expense of accuracy performance, which we will show in Section 4.5.

4.3.3 Rate-of-Change for Threshold Optimisation

4.3.3.1 CA-outlier

Let S be projected relation of n tuples S={s_1,…, s_n}. Given a threshold β, a **C**orrelation-based **A**ttribute (CA-)outlier is a paired set (A, s_i), 1≤ i≤ n such that the deviation scores of A w.r.t

s_i based on an outlier metric (O-measure or Q-measure) is less than β. Optimal value of β can be automatically derived using Rate-of-change.

4.3.3.2 Rate-of-Change

Given an attribute A and the set of O-measure(A, s_i) $\forall s_i \in S$, 1≤ i≤ n. Let L be the list of tuples s_i sorted in ascending order of O-measure(A, s_i). The Rate-of-change of a ranked tuple s_i (2≤i≤n) is defined as

$$Rate-of-change(s_i)=\frac{O-measure(A,s_i)-O-measure(A,s_{i-1})}{O-measure(A,s_{i-1})} \tag{4.3}$$

The same formula is applicable to determine the Rate-of-change based on the Q-measure metrics. Intuitively, the Rate-of-change measures the extent of increments in the outlier scores. The maximum Rate-of-change indicates the point at which the outlier scores increase suddenly and intuitively suggests the boundary between the outliers and the non-outliers.

4.4 Attribute Outlier Detection Algorithms

In Section 4.3.1.1, we discussed that a correlation-based attribute outlier (CA)-outlier is a paired set (A, s_i) of an attribute A and a tuple in the projected relation $s_i \in S$. Following that attribute outliers are defined with respect to a tuple in a projected relation (data subspace) S, we decompose the detection of CA-outliers into 2 sub-problems:

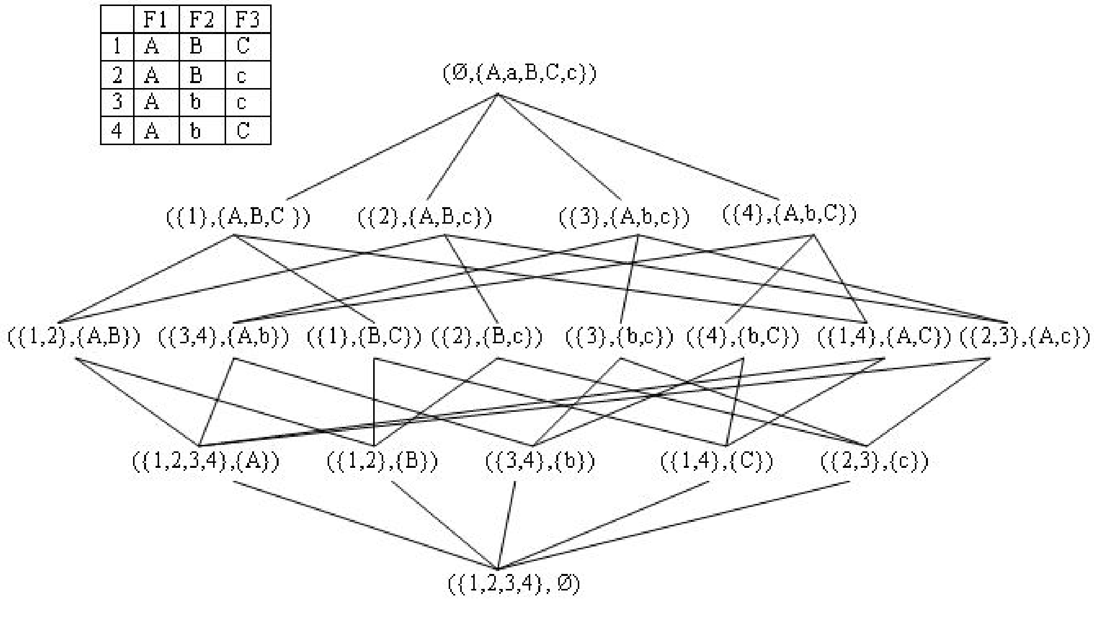
Step 1: Generate all possible projected relations and supports. The enumeration of the subspaces is equivalent to a *concept lattice*, where each node corresponds to a tuple s_i in a projected relation S.

Step 2: Calculate outlier scores and flag detected CA-outliers. The outlier scores are computed based on either the O-measure or Q-measure, depending on the nature of the input data and the efficiency requirements. Rate-of-Change determines the cut-off threshold to divide the outliers and non-outliers.

4.4.1 Subspace Generation using Concept Lattice

A concept lattice (also known as the Galois lattice) is a mathematical structure of a set of formal concepts, each comprising of the examples covered by the context (extension) and the descriptions of the concept (intension) [Wil82, GW99]. The enumeration of projected relations in Step (1) of the outlier detection process resembles the building a concept lattice.

Let us first formally define some of the notations used in concept lattice with respect to a database relation R with attributes $A = \{A_1, A_2, \ldots, A_m\}$. The concept lattice of a *formal context* L(R, A, C) describes a set of database tuples R, a set of attributes A, and a relation $C \subseteq R \times A$. Each node in L, known as a *formal concept* is a pair set (E, I), where extension $E \subseteq R$ and intension $I \subseteq A$. A partial order relation can be built on all concept lattice nodes. Given $l_1=(E_1, I_1)$ and $l_2=(E_2, I_2)$, let $l_1 < l_2 \Leftrightarrow I_1 \subset I_2$, the precedent order means l_2 is a direct parent of l_1 if I_2 is an attribute superset of I_1 and the latter is a sub-tuple of I_2. Figure 4.2 shows an example of the concept lattice generated from a relation of 3 attributes. Relating the concept lattice to our definitions in Section 4.3, each formal concept correspond to a tuple in the projected relation, and the degree of the attribute combination is the cardinality of the intension |I|, and the support is the cardinality of the extension |E|.



	F1	F2	F3
1	A	B	C
2	A	B	c
3	A	b	c
4	A	b	C

Figure 4.2: Example of a concept lattice of 4 tuples with 3 attributes F1, F2, and F3

Concept lattices are commonly used in frequent itemset mining for the derivation of association rules [HLS99, ZPOL97, Zak04]. Various algorithms have been proposed to improve the time efficiency of building a concept lattice. We will not discuss the time complexity of the lattice building step; a study of the performance of various algorithms is found in [KO02]. Rather, we focus on the more computationally demanding Step (2) and discuss a brute force as well as a pruned approach with reduced complexity.

4.4.2 The ODDS Algorithm

We first discuss the algorithm and the time complexity of the ODDS approach for CA-outlier detection.

4.4.2.1 Outlier Detection Algorithm

Algorithm 1 shows the details of the ODDS algorithm. A top-down iteration over the data subspaces, starting from the original relation R to the projected relations at degree 3 is performed [line 2]. The tuples or attribute combinations are stored into a list L. For each target attribute of each attribute combinations at degree k, the outlier scores (based on O-measure or Q-measure) are computed [line 6-8]. For each attribute, Get_CA-outliers function accepts a list of all attributes values of the same dimension and their corresponding O-measure or Q-measure values, and returns the detected CA-outliers.

Algorithm 1: ODDS

Input: Enumerated projections of degree 2..*m* for relation *R* with m attributes. User option of outlier metric.

Output: CA-outliers and the corresponding tuples of projected relations

```
1. List S, S' ← Ø
2. For degree k ← m to 3 do
3.       S ← projected relations of degree k and supports
4.       S' ← projected relations of degree (k-1) and supports
5.       For each s in S do
6.             For each attribute A of s do
7.                   $O_A$ ← add ComputeOutlierScores(A, s, S, S', sup(A), metric option)
8.             Endfor
9.       Endfor
```

```
10.     For each attribute $A$ of $R$ do            // compute Rate-of-change
11.         OUTPUT Get_CA-outliers ($O_A$)
12.     Enddo
13. Endfor
```

ComputeOutlierScores takes in an option of the outlier metric to use. Q-measure is less computationally costly; it is the quotient of the support of a subspace and the neighbourhood of the target attribute (conditional probability). On the contrary, O-measure adds an additional complexity to the computation [line 6-8] where the supports of every neighbour of the target attribute are computed. The extent of the additional computational burden is described in section 4.4.2.2.

Algorithm 2: ComputeOutlierScores

Input: Projections of degree *m-1* (*S'*) and target attribute (*A*) and subspace (*s*). User option of outlier metric.

Output: Outlier score

```
1.  If (metric is Q-measure) then
2.      Q-measure($A$, $s$)=GetSupport($S$, $s$)/GetSupport($S'$, $N(A, s)$)
3.      return Q-measure($A$, $s$)
4.  Else if (metric is O-measure) then
5.      O-measure($A$, $s$)=0
6.      For each neighbors $C_i$ of $A$ do
7.          O-measure($A$, $s$)=O-measure(A,s)+GetSupport(S', $N(C_i,s)$)
8.      Endfor
9.      O-measure($A$, $s$)=O-measure($A$, $s$)/ GetSupport($S'$, $N(A, s)$)
10.     return O-measure($A$, $s$)
11. Endif
```

In Get_CA-outliers, the input attribute points are sorted in ascending values of their outlier scores [line 1] to identify the maximum Rate-of-change [line 3]. Attribute points above max Rate-of-change are output as CA-outliers [line 6-8].

Algorithm 3: Get_CA-outliers (O_A)

Input: List of attributes A_j and subsets with O-measure or Q-measure values

Output: CA-outliers according to adaptive Rate-of-change thresholds

```
1.  B ← $O_A$ sorted in ascending order of measure($A_j$)
2.  For each point $b_i$, $2 \le i \le |B_j|$ do
3.      Rate-of-change($b_i$) = ($b_i$ - $b_{i-1}$)/ $b_i$ // rate of change
4.  Endfor
5.  ß ← i with max Rate-of-change($b_i$)
6.  For each $b_i$, $1 \le j \le$ ß do
7.      OUTPUT CA-outliers ← $b_i$
8.  Endfor
```

4.4.2.2 Time Complexity Analysis

The complexity of the algorithm is the complete search space of CA-outliers. Since a CA-outlier is a paired set (A, s_i) of an attribute A and a tuple in the projected subspace S, $s_i \in S$, the total number of possible values of a CA-outlier is the enumeration of attributes multiply by the total number of subspaces.

Lemma 1: Given a relation R with m attributes, the total number of columns in all possible projections of R is $m2^{m-1}$.

Proof. At each k (0≤ k≤ m), the total number of k-projections of m attributes is $\binom{m}{k}$. The total number of columns across all projections is the sum:

$$\begin{aligned}
&\sum_{k=0}^{m} k\binom{m}{k} = \binom{m}{1} + 2\binom{m}{2} + \ldots + (m-1)\binom{m}{m-1} + m\binom{m}{m} \\
&= m + 2\frac{m!}{(m-2)2!} + \ldots + (m-1).\frac{m!}{(m-m+1)!(m-1)!} + m\frac{m!}{(m-m)!m!} \\
&= m + m\left[\frac{(m-1)!}{((m-1)-1)!1!}\right] + \ldots + m\left[\frac{(m-1)!}{((m-1)-(m-2))!(m-2)!}\right] + m\left[\frac{(m-1)!}{((m-1)-(m-1))!(m-1)!}\right] \\
&= m\left[\binom{m-1}{0} + \binom{m-1}{1} + \ldots + \binom{m-1}{m-2} + \binom{m-1}{m-1}\right] \\
&= m\sum_{k=0}^{m-1}\binom{m-1}{k} = m2^{m-1}
\end{aligned}$$

Theorem 1: Given a relation R with m attributes and n tuples, the operation ODDS with Q-measure has a complexity of $\mathrm{O}\left(nm2^{m-1}\right)$.

Proof. Given Lemma 1, the total number of columns across all k-projections is $m2^{m-1}$. Since n is the maximum number of tuples across all projections, then $\mathrm{O}\left(nm2^{m-1}\right)$ is the worst-case total of all attributes in all subspaces.

Theorem 2: Given a relation R with m attributes and n tuples, the operation ODDS with O-measure has a complexity of $\mathrm{O}\left(nm(m+1)2^{m-2}\right)$.

Proof. To compute the O-measure outlier scores for each target attribute in *ComputeOutlierScores*, the support of every other neighbourhood of A is calculated. The number of iterations between line 6-8 of ODDS algorithm becomes

$$\sum_{k=0}^{m} k^2 \binom{m}{k} = \binom{m}{1} + 2^2 \binom{m}{2} + \ldots + (m-1)^2 \binom{m}{m-1} + m^2 \binom{m}{m}$$

$$= m^2 \left[\binom{m-2}{0} + \binom{m-2}{1} + \ldots + \binom{m-2}{m-2} \right] + m \left[\binom{m-2}{0} + \binom{m-2}{1} + \ldots + \binom{m-2}{m-2} \right].$$

$$= m(m+1) \sum_{k=0}^{m-2} \binom{m-2}{k} = m(m+1)2^{m-2}$$

Whither using O-measure or Q-measure, the brute-force approach of searching every data subspace of a relation for CA-outliers is highly inefficient.

4.4.3 Pruning Strategies in ODDS

To reduce the time complexity of Algorithm 1, we propose two filtering strategies to identify and prune data subspaces that cannot possibly contain an attribute outlier. This is illustrated in Figure 4.3, which shows part of the concept lattice generated from a relation of 5 arbitrary attributes, and the supports of each attribute combination. The numerical values at the top right corner of the combinations are the corresponding supports.

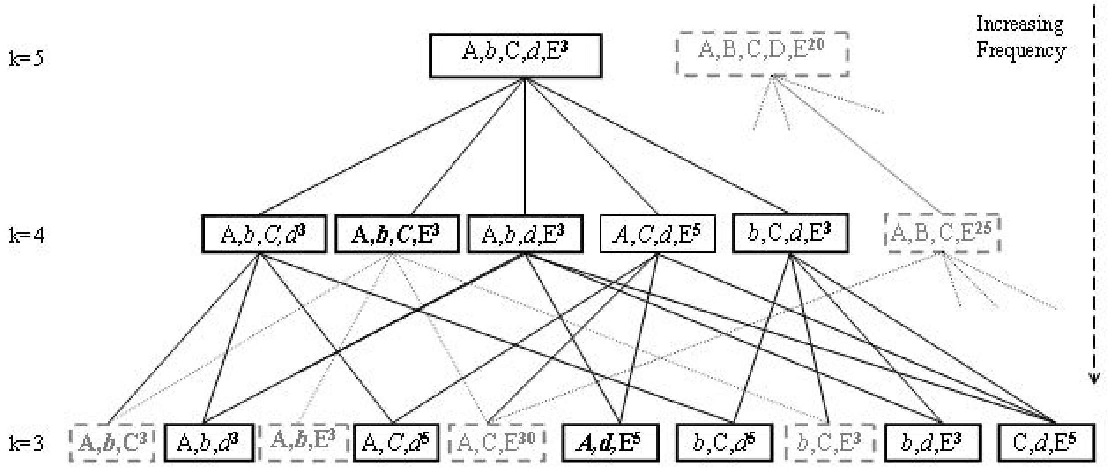


Figure 4.3: Attribute combinations at projections of degree k with two attribute outliers - b and d

We assume that all possible projections of the relation are completely enumerated. Intuitively, a frequent tuple of any projected relation cannot be a CA-outlier. Our first strategy filters any tuple s with sup(s) ≥ *minsup*, s and its sub-tuples from the calculation of the outlier scores. Pruning of sub-tuples follows the *Apriori property*: supports of sub-tuples increase across projected relations of decreasing degrees. For example, sup(<’A’,‘b’,‘C’,‘d’,‘E’>) ≤ sup(<’A’,‘C’,‘d’,‘E’>) ≤ sup(<’A’,‘C’,‘E’>). In Figure 4.3, setting *minsup* at 20 will prune off <’A’,‘B’,‘C’,‘D’,‘E’> with sub-tuples <’A’,‘B’,‘C’,‘E’> and <’A’,‘C’,‘E’>.

The second filtering strategy only applies to the Q-measure metric which exhibits the monotone property. We prove that if ‘*b*’ is a CA-outlier in a tuple s based on Q-measure, it is also CA-outlier in the sub-tuples of s.

Theorem 3: Let s be a tuple in projected relation S. An attribute A is a CA-outlier w.r.t s based on Q-measure implies that A is a CA-outlier w.r.t any sub-tuple of s which also contains A.

Proof. Let *b* be a CA-outlier w.r.t s=<’A’,’*b*’,’C’,’D’> detected using the Q-measure deviation metric. Let s' be a sub-tuple of s. Let β be the optimal threshold such that for any CA-outlier A, Q-measure(A, s)≤ β. Based on the *Apriori property*, we have

$$\sup(N(b,s)) \le \sup(N(b,s'))$$

$$Q-measure(b,s') = \frac{\sup(b)}{\sup(N(b,s'))} \le \frac{\sup(b)}{\sup(N(b,s))} = Q-measure(b,s) \le \beta$$

Hence, b is also a CA-outlier in s'. Sub-tuples of any CA-outlier found using Q-measure in an attribute combination are eliminated from deviation computation. In Figure 4.3, sub-tuples <‘A’,’*b*’,’C’>, <’A’,’*b*’,’E’> and <’*b*’,’C’,’E’> are omitted when ‘*b*’ is detected a CA-outlier in <’A’,’*b*’,’C’,’E’>. Beyond reducing the time complexity of the outlier score calculation, Theorem 3 enables reduction of the time for enumerating the projections.

4.4.4 The prune-ODDS Algorithm

Algorithm 4 shows the details of the prune-ODDS algorithm with Q-measure. A top-down iteration over the data subspaces, starting from the original relation R to the projected relations at level 3 is performed [line 3]. The tuples of attribute combinations are stored into a list *S*. Iteration begins by eliminating tuples which have supports greater than *minsup* from *S* [line 4]. Sub-tuples of degree (k-1) data subspaces are generated from the existing tuples in *S* at the beginning of each iteration [line 4].

Algorithm 4: prune-ODDS

Input: Enumerated projections of degree 2..*m* for relation *R* with m attributes. User option of outlier metric. User input *minsup*.

Output: CA-outliers and the corresponding tuples of projected relations

```
1.  List S, S' ← Ø
2.  S ← projected relations of degree m with supports ≤ minsup
3.  For degree k ← m to 3 do
4.      S' ← sub-tuples of S of degree (k-1) and supports ≤ minsup
5.      For each s in S do
6.          For each attribute A of s do
7.              $O_A$ ← add ComputeOutlierScores(A, s, S, S', sup(A), metric)
8.          Endfor
9.      Endfor
10.     For each attribute A of R do            // compute Rate-of-change
11.         OUTPUT $O_F$ ← Get_CA-outliers ($O_A$)
12.         Remove in S' sub-tuples of $O_F$
12.     Enddo
13.     S ← S'
14.     If S is empty then
15.         TERMINATE program
16.     EndIf
17. Endfor
```

The Q-measures are computed for each target attribute of each attribute combinations at degree k [line 6-8]. Based on theorem 3, the sub-tuples of CA-outliers are removed at line 12. The program terminates if the list *S* is empty [line 15].

4.5 Performance Evaluation

We evaluate the performance of ODDS on a synthetic data set and a real-world protein database. Experiments were performed on a Pentium-IV 1.8GHz computer with 2GB of main memory, and running Windows XP. Programs are written using a combination of Perl and C++.

4.5.1 World-Clock Data Set

The synthetic data set contains 9 attributes and 50,000 tuples generated from http://www.timeanddate.com/worldclock/. The original data set is free of any form of data noise, thus preventing the implicit noise in the original data set from interfering with the artificial noise introduced.

Table 4.2: Number of attribute outliers inserted into World-Clock data set

Attributes	X1	X2	X3	X4	Mix3
Country	311	299	298	287	295
State	311	320	294	286	315
City	310	339	302	322	336
Day	83	94	106	184	96
Time	278	271	262	285	374
Sunrise	337	311	305	291	316
Sunset	333	308	317	296	325
Postal Code	299	312	315	286	318
Continent	238	246	300	263	225

In order to evaluate the performance of ODDS at varying numbers of attribute outliers per tuple, we introduce *x* artificial attributes outliers to a random tuple in the data set. These attributes are assigned random values from their respective domains. The four datasets containing x=1, 2, 3, and 4 outliers per tuple are denoted X1, X2, X3 and X4 respectively. For example, X2 has 2,500 CA-outliers (5%) distributed across 1,250 tuples, each containing 2 attribute outliers. We also generate a Mix3 dataset by randomly inserting 1 to 3 artificial attribute outliers to each randomly selected tuple. Table 4.2 shows the number of attribute outliers inserted into World-Clock data set.

4.5.1.1 Accuracy at varying number of outliers per tuple

The accuracy of ODDS depends on the effectiveness of the outlier metric as well as the Rate-of-change. The maximum Rate-of-change is the point where the outlier scores change significantly. Figure 4.4 shows the Rate-of-change scores of each attribute derived from the ODDS/O-measure. The maximum Rate-of-Change for each attribute matches the number of outliers inserted in Table 4.2, and an F-score of 100% is achieved for X1 (Table 4.3). This indicates not only is the Rate-of-change effective in determining the optimal cut-off thresholds differentiating the outliers (positives) from the non-outliers (negatives), O-measure also accurately quantifies the extent of deviation of the attribute outliers. Subsequent experiments utilize the Rate-of-change factors as default selection for thresholds.

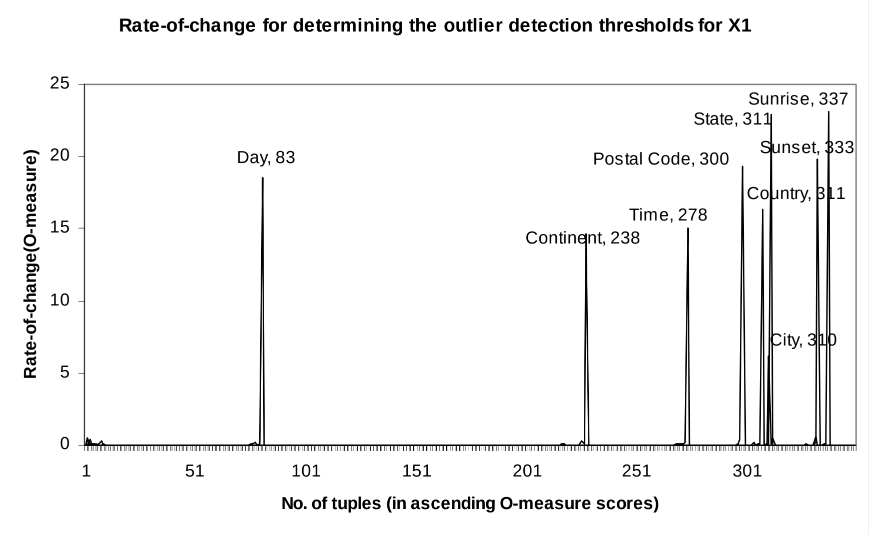


Figure 4.4: Rate-of-change for individual attributes in X1

Table 4.3 shows the F-scores of ODDS at varying number of CA-outliers per tuple. With only 9 attributes, it is not surprising that the false-negatives escalate when tuples contain 4 or more CA-outliers per tuple. For data sets containing between 1 to 3 attribute outliers in each tuple, the outlier detection method can achieve an F-score of between 73% and 100%. Precision is generally high (92-100%), meaning that the FP rate is low. We expect that real-world data set will contain a mixture of different number of attribute outliers in each tuple. For this, ODDS/O-measure achieves an F-score of 88% for the Mix3 data set.

Table 4.3: Performance of ODDS/O-measure at varying number of CA-outliers per tuple

	Recall (%)	Precision (%)	F-score(%)
X1	100	100	100
X2	90	100	95
X3	63	99	73
X4	39	92	50
Mix3	79	99	88

4.5.1.2 Convergence across projections

In reality, we do not know the number of attribute outliers that may be present in each tuple of a database. The ODDS approach systematically searches for CA-outliers, identifying tuples with only one outlier at the data subspaces of the highest degree k (i.e. complete tuple), and others at the subsequent lower degree projections. The detected CA-outliers accumulate across the projections.

The Mix3 data set is used to evaluate the performance of ODDS algorithm using O-measure and Q-measure metrics respectively. The accuracy of ODDS converges across the projected relations of degree k, starting from k=7, with decreasing false negatives as the number of attribute outliers detected accumulate. Figure 4.5 shows the F-score is between 70% to 88% with O-measure and Q-measure.

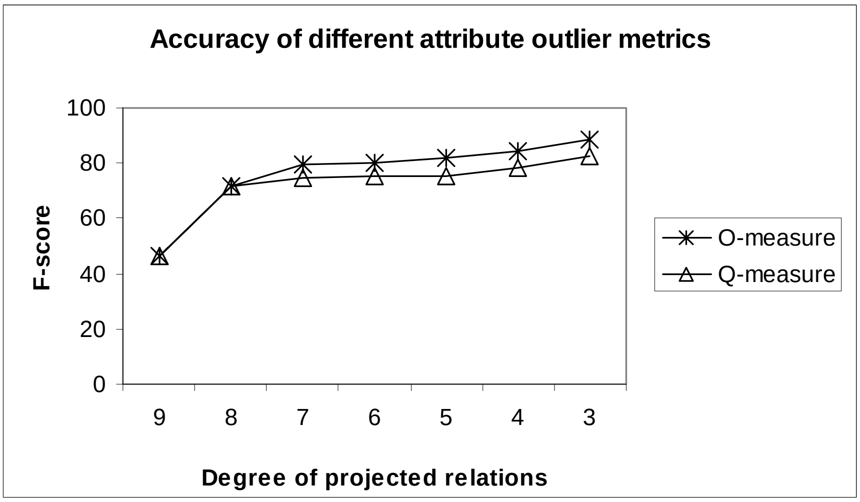


Figure 4.5: Accuracy of ODDS converges in data subspaces of lower degrees in Mix3

4.5.1.3 Comparison with other methods

ODDS/O-measure and ODDS/Q-measure perform consistently better than classifier-based methods using decision tree C4.5 [ZW04, Ten04]. Its performance is also stable when the percentage of outlier noise increases. As the percentage of attribute outliers in the data set increases, the correlations between attributes decreases, thus affecting the accuracy of the correlation-based outlier detection approach.

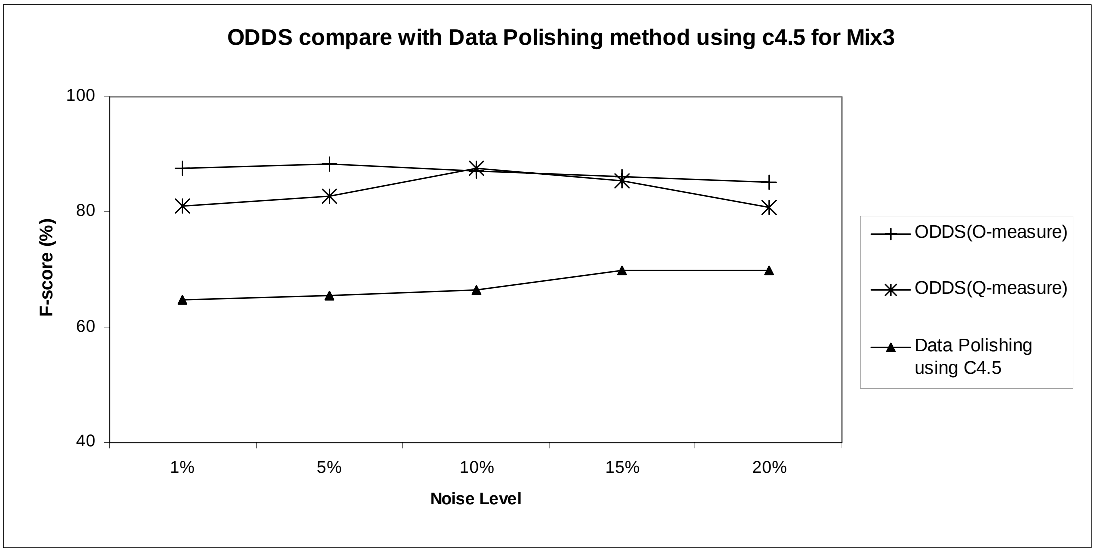


Figure 4.6: Performance of ODDS compared with classifier-based attribute outlier detection

4.5.2 UniProt Protein Data Set

The UniProt database (release 7.1) consisting of 2,826,395 protein sequence records are collected from multiple sources of large-scale sequencing projects and is frequently accessed by the world-wide biological researchers for analysis and data mining. As discussed in Chapter 3 (Section 3.3.3.1), UniProt/TrEMBL records are computationally annotated, thus the protein functions are predicted rather than verified experimentally, they contain a significant portion of annotation errors. UniProt/TrEMBL contains primarily free-text and is highly sparse. For this experiment, we focus on 5 attributes relevant to the protein annotated properties.

Table 4.4: Description of attributes in UniProt

Attribute	Distinct values	Multiple values	Description
OR	6	No	Organism source of the protein
KW	898	Yes	Keywords subject reference for the protein
GO	8,486	Yes	Gene ontology controlled vocabulary of proteins' properties.
PN	669,151	No	Proposed official name of protein
SY	126,299	Yes	List of synonyms of the protein

Table 4.4 shows that the protein name PN, gene name GN, synonym SY each contain more than 100,000 unique values. These large numbers suggest the high sparse-ness of the UniProt database. In fact, the naming of proteins and genes are often left to the discretion of the experimentalists who submit these sequences into the database, hence, a large percentage of these names are rare but legitimate.

Table 4.5: Frequencies of GO target attributes identified at projections of degree 5 of UniProt data set

Attribute value	Freq	Attribute value	Freq
ATP_binding	172,090	iron ion binding	81,693
kinase activity	30,355	antigen processing, exogenous antigen via M..	6,786
binding	27,933	membrane	295,807
calcium ion binding	13,481	metabolism	87,071
catalytic activity	50.693	methyltransferase activity	7,905
chromosome segregation	1,205	negative regulation of cytokine production	1
cytokine activity	816	phosphoprotein phosphatase activity	1,693
defense response to bacteria	408	positive regulation of MHC class II biosynt…	19
drug transporter activity	1,304	protein-tyrosine kinase activity	3,817
drug transport	189	protein amino acid phosphorylation	32,462
electron transporter activity	1	protein binding	42,461
extracellular space	5,133	protein biosynthesis	41,101
hydrolase activity	71,025	response to unfolded protein	6,311
integrase activity	2,230	ribosome	17,865
interferon-gamma biosynthesis	15	signal transducer activity	10.702
signal transduction	23,060	spindle pole body	84
transferase activity	114,389	structural constituent of ribosome	25,059
translational initiation	2,416	translation initiation factor activity	3,747
Intracellular	65,544	transporter activity	57,341

A preliminary experiment applying the ODDS/O-measure to the UniProt data set reveals that the O-measure metric does not distinguish the rare attributes from the attribute outliers. Table 4.5 shows the frequencies of the target attributes (neighbourhoods not shown) in the GO dimension identified from projected relations of degree 5 of the data set. We notice that some of the outliers detected are in fact rare attribute values, including “negative regulation of cytokine production”, “electron transporter activity”, “interferon-gamma biosynthesis”, “positive regulation of MHC class II biosynt…” each with a frequency of less than 20 in the data set.

Since we are not interested to detect these rare attribute values, we adjust the O-measure metric with a frequency factor to penalize the metric by the frequency of the target attribute; the new metric is called O$_f$-measure.

4.5.2.1 O$_f$-measure

Let freq(A$_v$) be the frequency of an attribute A$_v$ in the original relation R. The O$_f$-measure of A$_v$ w.r.t a tuple s is defined as

$$O_f - measure(A_v, s) = \frac{\sum_{i=u}^{v-1} \sup(N(A_i, s))}{\sup(N(A_v, s)) freq(A_v)} \tag{4.4}$$

O$_f$-measure takes into account the support of A$_v$ in R. A lower weightage is given to the rare attribute values with lower frequencies. Unlike O-measure, O$_f$-measure favours non-rare values and is more effective in identifying attribute outliers in sparse data set which contained vast occurrences of rare attribute values which are not attribute outliers. In certain sparse data sets such as the UniProt database, finding the vast occurrences of rare attribute values such as ‘B.C’ (Table 4.1), which is not necessarily erroneous is not of prime interest.

As an example, consider the attribute outlier *X* in Table 4.1. Given the low frequency of the value ‘B.C.’ in the data set, the low O-measure score almost guarantee that the State attribute in s= <’Canada’, ‘B.C.’, ‘Vancouver’> will be labelled as an outlier, that is, O-measure(State, s)=(1+1)/4=0.5. In contrast, O$_f$-measure(State, s) = (1+1)/(4*0.09) = 5.6 is relatively much higher.

4.5.2.2 *CA-outliers in UniProtKB/TrEMBL*

We applied ODDS/O_f-measure to the complete UniProt data set. Table 4.6 shows the number of outliers detected for each attribute. CA-outliers are redundantly identified across different degree of the projected subspaces; the accumulated number of affected records shows the non-redundant number of affected records.

Table 4.6: CA-outliers detected in UniProtKB/TrEMBL using ODDS/O_f-measure

CA-outliers detected at projections of degree	OR	PN	KW	GO	SY
3	27 (73)	45 (24)	56 (31)	17 (97)	18 (5)
4	333 (553)	136 (6033)	276 (196)	378 (2196)	186 (124)
5	195 (45)	40 (13)	57 (17)	308 (2365)	132 (56)
Accumulated	(671)	(6070)	(241)	(2365)	(185)

* Brackets contain number of affected records.

4.5.2.3 *Verification of CA-outliers detected*

Biologists through manual verification check the validities of CA-outliers found in the GO dimension. **True positive TP** indicates an uncommon association of the target attribute with the other attributes in the projected relation. **False positive FP** indicates that no peculiarity is found in the correlation behaviour of the target attribute. **Indeterminable** means that further investigation is required.

The manual verification step largely depends on the knowledge level of the biologist and his decisive-ness. Table 4.7 shows that a large percentage (24%-46%) of the CA-outliers require further investigation because the biologist lacks the detail knowledge to justify if the annotation is erroneous or it is only exceptional. 27%-58% are false positives. 10%-24% (or 19%-55% positive predictive rate among those justifiable) of the gene ontology attribute outliers are confirmed result of erroneous annotations.

The experiment shows that ODDS can be used as a pre-step for cleaning protein annotations, subjected to further verification by an annotator. Obvious cases of erroneous annotations are found in the ODDS results. For example, 12 bacteria proteins (Q9Z5E4, Q6J5G7, among others) are associated with viral capsids which are protein coats for viral particles. Also, 5 eukaryote proteins (Q9BG87, Q4IJ15, among others) are oddly related to the reproduction of viruses.

Table 4.7: Manual verification of Gene Ontology CA-outliers detected in UniProtKB/TrEMBL

Annotation	CA-outliers	TP	FP	Indet.
CA-outliers detected at 3-attribute projections	17	6	5	6
CA-outliers detected at 4-attribute projections	378	65	221	92
CA-outliers detected at 5-attribute projections	308	31	136	141

4.6 Concluding Section

Existing outlier detection methods focus primarily on class outliers; limited research has been conducted on attribute outliers. This work presents a novel method called ODDS that utilizes the correlations between attributes to identify attribute outliers. Rather than focusing on rare attribute values or rare classes, ODDS systematically searches for attribute points that exhibit alternative correlation behaviour when compared to other attribute points in a data subspace. These local deviators which we refer to CA-outliers are bivariate. Experimental evaluation shows that ODDS can achieve F-score of up to 88% in synthetic data set and is practically applicable for detecting erroneous annotations in a protein database.

This chapter focused on the accuracy of the outlier detection approach. Two filtering strategies are applied to reduce the running time of the ODDS algorithm where the enumeration of data subspaces is a major bottleneck. To reduce the running time further, one strategy is to separate the data space into partitions of correlated subspaces in order to reduce the number of enumerated projections. This idea is evaluated in the next chapter which proposes the use of XML hierarchical structures to derive meaning partitions, thus improving the effectiveness and efficiency of attribute outlier detection.

Chapter 5: Attribute Outlier Detection in XML using XODDS

Although nature commences with reason and ends in experience, it is necessary for us to do the opposite, that is to commence with experience and from this to proceed to investigate the reason.

Leonardo Da Vinci
Engineer, Painter, & Sculptor (1453 - 1519)

Despite the recent proliferation of semi-structured data models such as XML as the standardized data representation in various domains, including bioinformatics, the development of data cleaning approaches for such data is at rudimentary stage. Even among the limited data cleaning works in XML discussed in the data cleaning survey in Chapter 2, the problem of duplicates is addressed, but not outliers. Existing outlier detection methods remain focus on relational data and they are not easily adaptable to XML data because of the inherent differences in data structures.

In this chapter, we propose a systematic 4-steps framework for outlier detection in XML data called **XODDS** (for **X**ML **O**utlier **D**etection from **D**ata **S**ubspace). Similar to the ODDS method, the XODDS framework utilizes the correlations between attributes to adaptively identify attribute outliers. In addition, XODDS leverages on the hierarchical structure of the XML document to provide contextual information lacking in relational data, with the aim of improving both the effectiveness as well as efficiency of identifying attribute outliers in XML documents. Specifically, we introduce two novel concepts of *correlated subspaces* and *aggregate attributes* in XML. The notion of correlated subspaces reduces the time complexity of the attribute outlier method by separating the XML document into several natural partitions according to the hierarchical structure. Aggregate attributes enables summarization of group of nodes, and thus facilitates data cleaning at higher level of abstractions. We also devise 6 key properties that a good metric for attribute outliers should satisfy, and compare other correlation-based measures to the *x*O-measure and *x*Q-measure metrics used in XODDS.

Experiments results on both synthetic and real-world data sets indicate that XODDS is effective in detecting attribute outliers in XML. In the detection of annotation errors in UniProt/TrEMBL, XODDS attains 97% positive predictive value (PPV) - with significant improvement over ODDS.

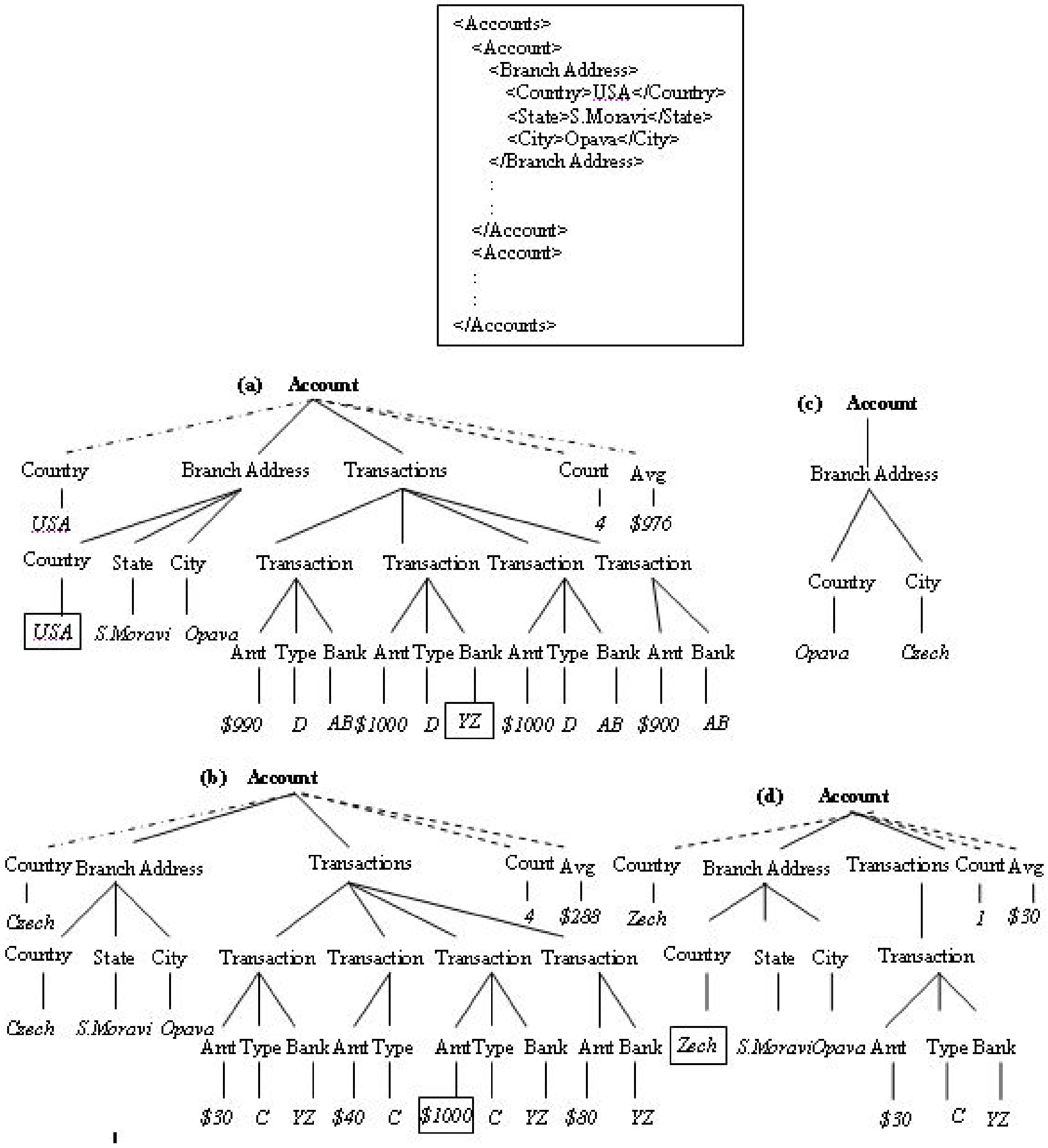
5.1 Introduction

In the recent years, increasing number of databases are converted into XML formats to facilitate data exchanges and publishing on the Web. For example, major biological databases have started to support XML formats; they include UniProt (UniProt XML) [WAB$^+$06], GenBank (NCBI XML, INSDSeqXML, and TinySeqXML) [BKL$^+$06], and PDB (mmCIF, and PDBML) [DAB$^+$05]. Regardless of this growth, current data cleaning methods focus on relational data. Although there are a few data cleaning works on XML, they mainly focused on duplicate detection [LTLL03, WN05, PWN06] (details in Section 2.5). On the contrary, the development of outlier detection methods for XML documents has been overlooked.

Intrinsic differences between XML and relational data models inhibit direct adaptation of conventional data cleaning methods to XML data model. XML data are often "sparse"; missing information are represented using optional elements. In contrast, relational data are usually "dense" and missing attribute values are represented by null values. This introduces bias to the XML data, which in turn falsely amplifies the magnitude of outlier-ness. Data sparseness should be taken into account when selecting a suitable metric for attribute outlier detection - a crucial step in attribute outlier detection. There exist numerous interesting-ness measures for ranking the usefulness and significance of the discovered co-relations or co-occurrences [HH01, TKS02]. However, they are primarily devised for the mining of association rules or for feature selection, and may not be naturally suited to the attribute outlier detection problem. In order to systematically compare the suitability of the two metrics used in XODDS - *x*O-measure and *x*Q-measure to the existing interesting-ness metrics, we develop 6 key properties of a good attribute outlier metric and evaluate the extent to which each metric satisfies these properties (details in Section 5.4).

Also, XML data contain multiple levels of nested elements (or attributes), whereas relational data are “flat”. Such encoding of relationships between the XML elements in the form of the hierarchy provides additional contextual information to the attribute outlier detection problem. Specifically, we utilize the hierarchical structures of XML to derive *correlated subspaces* and *aggregate attributes* that advance the outlier detection process. We use a concrete Bank account example often used for fraud detection to elaborate these two concepts.

5.1.1 Motivating Example



```
<Accounts>
  <Account>
    <Branch Address>
      <Country>USA</Country>
      <State>S.Moravi</State>
      <City>Opava</City>
    </Branch Address>
      :
      :
  </Account>
  <Account>
  :
  :
</Accounts>
```

Figure 5.1: Example bank accounts XML document

Consider the 3 attribute outliers shown in Figure 5.1. The <Country:USA> element in Figure 5.1(a) is erroneously associated with the <State:South Moravi> and <City:Opava> of Czech Republic. <Bank:YZ> is unusual because most transactions from this account are credited to

Bank AB. In Figure 5.1(c), a transaction involving $1000 is unusual because transactional amounts for this account are typically less than $100.

First, we observe that attribute outlier-ness is a bivariate property of a target attribute and a localized neighbourhood of correlated attributes. By inspecting the correlations across all <Country>, <State> and <City> elements in Figure 5.1(a), we can deduce that <Country:USA> is an attribute outlier from its negative correlations to <State:S.Moravi> and <City:Opava>. Conversely, comparing the same Country element to <Amt:$1000> or <Bank:AB> make little sense in identifying attribute outliers. The observation is in agreement to our definition of a correlation-based attribute outlier in Chapter 4.

A correlation-based attribute outlier is an object which exhibits abnormal behaviour in a subspace of related objects.

Notice that the structure of XML provide hints of what constitutes "a subspace of related objects". The notion of *correlated subspaces* in XML leverages on the nested structure to identify groups of logically correlated elements. For example, the elements <Country>, <State> and <City> form a correlated subspace, that describes the <Branch Address> of any account. The <Amt>, <Bank> and <Type> elements of <Transaction> elements in Figure 5.1(a) form a subspace, which is however different from the transactional subspace in Figure 5.1(b). Realistically, each account has different spending power and therefore transactions are not comparable across different accounts.

Figure 5.1 also shows the *aggregate attributes* (indicated by dotted lines) that have been derived from aggregate functions such as AVG, COUNT, MIN, MAX, and SUM to summarize sub-structures in XML through computing a scalar value from a set of multiple elements [GHQ95, GCB+97]. Certain XML query language such as XQuery and XML-GL support aggregate attributes [CCD+99, CFR+01]. Aggregate attribute can facilitate the identification of outliers at higher level of abstractions. For example, through the correlation patterns of the <Count>, <Avg> and <Country> elements in Figure 5.1(a)-(d), we can identify accounts with unusually low (or high) transactional averages or counts, compared to other accounts from the same country.

Logically separating the Bank Account XML (details of computational derivations in Section 5.2) generates the 4 subspaces shown in Figure 5.2. The correlation behaviour of each attribute is analysed with respect to individual subspaces, rather than the entire XML document. Consider the subspace in Figure 5.2(c), evidence that <Country:USA> is an attribute outlier can be drawn from the correlation behaviour of <Country:USA> with <State:S.Moravi> and <City:Opava>. This observation motivates the development of *x*O-measure and *x*Q-measure. The same deduction identifies "Zech", which is, however a rare attribute rather than an attribute outlier. In Chapter 4, we explained that rarity does not equate to outlier-ness, and individual elements with rare values are not necessarily outliers. The transactions of each account are organised into different subspaces in 5.2(a) and 5.2(b); intuitively, we are looking for deviation in transactional behaviour of individual account holders, and not across. Also notice that the aggregate attributes are organised into a separate subspace which is useful for tracking the transactional behaviour across different countries.

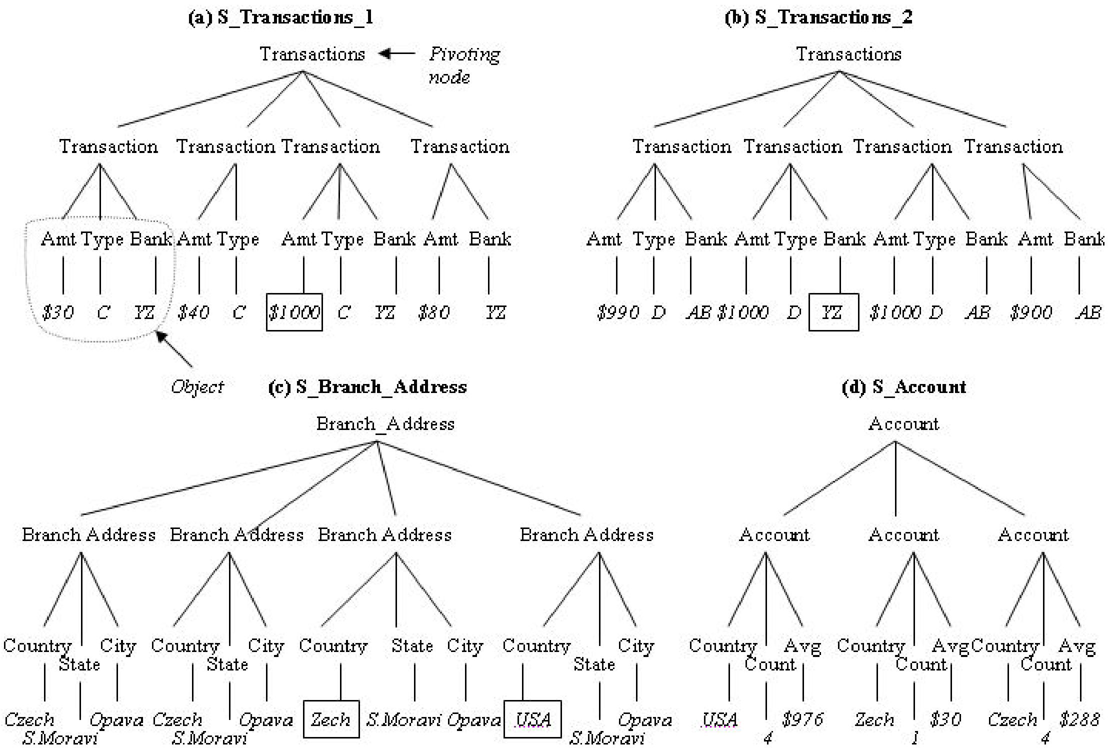

(a) S_Transactions_1
Transactions
Pivoting node
Transaction Transaction Transaction Transaction
Amt Type Bank Amt Type Amt Type Bank Amt Bank
$30 C YZ $40 C $1000 C YZ $80 YZ
Object
(b) S_Transactions_2
Transactions
Transaction Transaction Transaction Transaction
Amt Type Bank Amt Type Bank Amt Type Bank Amt Bank
$990 D AB $1000 D YZ $1000 D AB $900 AB
(c) S_Branch_Address
Branch_Address
Branch Address Branch Address Branch Address Branch Address
Country State City Country State City Country State City Country State City
Czech S.Moravi Opava Czech S.Moravi Opava Zech S.Moravi Opava USA S.Moravi Opava
(d) S_Account
Account
Account Account Account
Country Count Avg Country Count Avg Country Count Avg
USA 4 $976 Zech 1 $30 Czech 4 $288

Figure 5.2: The 4 subspaces in Bank Account XML

The rest of the chapter expand on the two concepts of *correlated subspaces* and *aggregate attributes* to devise a complete framework for detecting attribute outliers in XML documents.

5.1.2 Contributions

Specifically, this chapter makes the following contributions:

1. We introduce the notion of **correlated subspaces,** which is a meaningful grouping of correlated objects based on the XML data structures.
2. We propose the use of **aggregate attributes** as summarizing elements in the hierarchical XML structures. The use of aggregate attributes enables the detection of attribute outliers at higher level of abstraction.
3. We develop two correlation-based attribute outlier metrics for XML, namely the ***x*O-Measure** and ***x*Q-Measure**. We identify 6 key properties that a good attribute outlier metric should satisfy, and use them to evaluate *x*O-Measure and *x*Q-Measure with other existing interesting-ness measures.
4. We develop a complete, systematic framework for detecting attribute outliers in XML called **XODDS** (for **X**ML attribute **O**utlier **D**etection from **D**ata **S**ubspaces) and demonstrate its effectiveness on real world datasets.

The rest of this chapter is organized as follows. Section 5.2 presents formal definitions used in XODDS and Section 5.3 describes the XODDS framework. Section 5.4 presents a comparative analysis of attribute outlier detection metrics. An experimental evaluation of XODDS is given in Section 5.5, and we conclude in Section 5.6.

5.2 Preliminary Definitions

First, we present formal definitions of the terms used in XODDS. An XML document can be modelled as a tree $T(V, E, r, L)$ where V is a set of n nodes, E is a set of edges $E \subseteq V \times V$ and r is the root node. Each node represents an element or an attribute.

Let L be a countable set of labels. The labelling function *label*: $V \rightarrow L$ maps each $v \in V$ to some $l \in L$; different nodes may have the same label. Each $e \in E$ is an ordered pair of nodes, $e = (v_i, v_j)$ where $v_i \in V$ is the *parent* of $v_j \in V$, and v_j is the *child* of v_i.

We use the function *value*(v) to denote the value of a leaf node. It follows that if v_j is a leaf node, value(v_j) ≠ Ø. For every node $v \in V$, there is a unique path from root node r to v, denoted by $p_{r,v} = (v_o = r, v_1 \ldots, v)$. The number of edges from r to v is *dist*(r, v). Without loss of generality, we say that r is an *ancestor* of v, or v is a *descendant* of r.

Consider the example in Figure 1(c). Let T be the tree rooted on Account node. L = {"Account", "Branch Address", "Country", "State", "City"}. <Country>, <State>, and <City> are child nodes of <Branch Address>, and their distances from root node are 2.

5.2.1 Correlated Subspaces

5.2.1.1 Object

Definition 1: Given a tree $T(V, E, r, L)$, an *object Obj*(v_i) *rooted at node* $v_i \in V$ is a set of nodes $v \in V$ such that *dist*(v_i, v)=1 and value(v)≠Ø. Simply, an object is denoted by its children leaf nodes. For example, the XML document in Figure 5.1(a) comprises of 3 objects. The <Branch Address> node forms an object with children nodes of <Country>, <State> and <City>. Similarly, Figure 5.2(a) shows *Obj* (<Transaction>) = {<Amt:$30>, <Type:C>, <Bank:YZ>}.

5.2.1.2 Nearest Common Ancestor

Definition 2: Given two nodes, v_i ,$v_j \in V$, $v_c \in V$ said to be the *common ancestor* of v_i and v_j if they are both descendents of v_c. v_c is called the *nearest common ancestor* of v_i and v_j, denoted as *NCA*(v_i ,v_j) $\in V$ if the distance between v_i and v_j through v_c is shorter than any $v \in V$. Consider the example in Figure 5.1(c). Let T be the tree rooted on Account node. L = {"Account", "Branch Address", "Country", "State", "City"}. <Country>, <State>, and

<City> are child nodes of <Branch Address>, and their distances from root node are 2. The nearest common ancestor *NCA*(<Country>,<City>)=<Branch Address>.

5.2.1.3 Correlated Subspace

Definition 3: Given a tree *T*(*V*, *E*, *r*, *L*), a node $v_p \in V$, which we call the *pivoting node* of the subspace, and $l_s \in L$, a correlated subspace *S*(v_p, l_s) is a set of nodes such that ∀ $v \in S(v_p, l_s)$

1. $Obj(v) \neq Ø$
2. for any v_i, $v_j \in S(v_p, l_s)$, $NCA(v_i, v_j)= v_p$
3. for any v_i, $v_j \in S(v_p, l_s)$, $p_{v_p,v_i} = p_{v_p,v_j}$

For example, the S_Transactions_1 and S_Transactions_2 in Figure 5.2 are subspaces, with l_s=”Transaction” and pivoting node v_p=<Transactions>. Its follows that the correlated subspace of Branch Address, S_Branch_Address = S(<Accounts>, “Branch_Address”). In general, *a correlated subspace is a container of comparable objects, and each object is represented by a set of leaf nodes*.

5.2.2 Aggregate Attributes

The contextual complexity of aggregate attributes requires that they are defined by the users rather than determined by the system. Such specifications can be facilitated through a user interface for the selection of aggregate functions and the nodes to aggregate. For example, in Figure 5.1(a) and (b), the <Transactions> node is summarized through computing the COUNT function on <Transaction> nodes and AVG aggregate function on the <Amt> nodes.

5.2.2.1 Aggregate Attribute

Definition 4: Given *T*, an aggregate attribute $v_a(v_i, p_{v_i,v}, F) \in V$ is a sibling leaf node of v_i derived from applying the aggregate function *F* on all descendent nodes of v_i following the path location $p_{v_i,v}$. In Figure 1(a), let *F* = AVG be applied to all <Amt> nodes following the xpath location ‘Transactions/Transaction/@Amt’. This generates the <Avg:$976> node which is inserted into the XML as a sibling node of <Transactions>.

5.2.3 Correlation Neighbourhood

Correlation-based attribute outlier-ness is a bivariate property of both the target attribute (leaf node) and a *correlation neighborhood*, which is a group of attributes that co-occur with the target attribute in the same subspace.

5.2.3.1 Correlation Neighbourhood

Definition 5: Given a node $v_i \in V$, $Obj(v_i) \neq \emptyset$. Let $V_s^k \subseteq Obj(v_i)$ be a subset of leaf nodes of the object defined over v_i where k is called the degree of V_s^k. For each target node $v_t \in V_s^k$, the *correlation neighborhood* of v_t in V_s^k is defined $N(v_t, V_s^k) = \{v \in V_s^k \mid v \neq v_t\}$. A neighborhood $N(v_t, V_s^k)$ is of *degree k* where k=$|N(v_t, V_s^k)|$. In Figure 5.2(c), the 2-degree neighborhoods of <State:S.Moravi> is the set of 2 nodes {<Country:USA>, <City:Opava>}.

5.2.4 Outlier Scoring

Outlier-ness is not a binary property. Simply labeling an object as an outlier serves little value; the objects have different degree of deviation, giving rise to the notion of strong or weak outliers. In the XML model, an attribute outlier metric is a measurement of the strength of the outlier-ness of each object in its correlated neighborhood. We adapt the O-measure and Q-measure in ODDS for the XML models, and call them *x*O-measure and *x*Q-measure.

5.2.4.1 Support

Definition 6: Given a subspace $S(v_p, l_s)$ and a node $v_i \in S(v_p, l_s)$, let $V_s^k \subseteq Obj(v_i)$. The support of V_s^k, denoted *sup*(V_s^k) is the count of the number of nodes $v \in S(v_p, l_s)$ such that $\forall\ v_s \in V_s^k$, there exists $e=(v, v_j) \in E$, $value(v_j)=value(v_s)$. In general sense, the support of a set of nodes V_s^k is the count of objects in the same subspace, with leaf nodes that have the same values as V_s^k. For example, the support of *sup*(*N*(<State:S.Moravi>))= sup({<Country:Czech>, <City:Opavi>}) = 2 in Figure 5.2(c).

5.2.4.2 xO-measure

Definition 7: Given *S*(*v*$_p$, *l*$_s$) and a node $v_i \in S(v_p, l_s)$, let $V_s^k \subseteq Obj(v_i)$. The x*O-measure* of a node $v_s \in V_s^k$, denoted O-measure(*v*$_s$, V_s^k) is defined as

$$xO - measure(v_s, V_s^k) = \frac{\sum_{v_j = v_i} \sup(N(v_j, V_s^k))}{\sup(N(v_s, V_s^k))} \tag{5.1}$$

where $v_j \in V_s^k, v_j \neq v_s$

*x*O-measure is basically the quotient of the co-occurrences of an attribute v with its neighbors and the co-occurrence of its neighbours. For example in Figure 5.2(c), the *x*O-measure of <Country:USA> in {<State:S.Moravi>, <City:Opava>} = ¼ and this is comparatively lower than *x*O-measure(<Country:Czech>, {<State:S.Moravi>, <City:Opava>}) = 2/4 = 0.5.

5.2.4.3 xQ-measure

Definition 8: Given *S*(*v*$_p$, *l*$_s$) and a node $v_i \in S(v_p, l_s)$, let $V_s^k \subseteq Obj(v_i)$. The *x*Q-measure of v in neighborhood $N(v_t, V_s^k)$ is defined as

$$xQ - measure(v_s, V_s^k) = \frac{\sup(v_s)}{\sup(N(v_s, V_s^k)} \tag{5.2}$$

Essentially, the *x*Q-measure is the conditional probability of a target attribute *v*$_s$ over its neighborhood. In Figure 5.2(b), the *x*Q-measure(<Bank:YZ>, {<Amt:$1000>, <Type:C>})=1/2=0.5. Notice that for both xO-measure and xQ-measure, *lower scores implies higher degree of attribute outlier-ness*.

The output from the outlier scoring step is a list of metric scores with respect to the node and its neighborhoods. For each attribute across neighborhoods of a projection degree k, the outlier scores are ranked in ascending order.

5.2.5 Outlier Identification

Outlier-ness scoring measures the objects according to their extent of outlier-ness, they do not differentiate between outliers and non-outliers. To isolate the attribute outliers from non-outliers, users typically need to define a threshold. This is not viable, given that the number of attribute outliers differ across different XML documents, and across different types of elements. In the XODDS algorithm, the optimal threshold is determined from the maximal *Rate-of-change* which is the data-dictated outlier score separating the outliers and non-outliers, thus removing the dependency of the outlier detection on any user-specified parameter. We compare by experiments the effectiveness of *Rate-of-Change* with the *Top-k* approach that selects a certain top percentage of the data set as outliers.

5.2.5.1 Rate-of-Change

Definition 9: Given a node and the set of *x*O-measure(v_s, V_s^k) at degree k and where *label*(v$_S$)=*l$_s$*, let *x*O-measure(v_s, V'^k_s) be the list sorted in ascending order of *x*O-measure(v_s, V_s^k). The Rate-of-change of a ranked object v_s (2≤i≤n) is defined as

$$Rate-of-change(v_s) = \frac{xO-measure(v_s, V'^k_i) - xO-measure(v_s, V'^k_{i-1})}{xO-measure(v_s, V'^k_i)} \qquad (5.3)$$

5.3 Outlier Detection Framework

The complexity of the outlier detection problem cannot be resolved in a single computational step. Rather, it demands streamlining of various algorithmic components into a systematic framework.

5.3.1 XODDS Framework

The XODDS framework is a generalized approach towards detecting attribute outliers from XML documents. Figure 5.3 details the specification requirements and processes in XODDS. There are four key steps:

1. **Attribute aggregation** is a specification step for users to define summarizing elements as well as where to insert them in the XML documents. Aggregate attributes serve as abstraction to the nested elements and enable the identification of outliers at higher levels of the abstraction.
2. **Subspace identification** determines semantically meaningful subspaces in the XML document. Deriving the subspaces requires identifying clusters of objects forming a subspace.
3. **Outlier scoring** computes the outlier scores for each object in its subspaces using *x*O-measure and *x*Q-measure. The support-based object pruning step reduces the enumeration of neighborhoods for outlier scoring and therefore the computational complexity.
4. **Outlier identification** distinguishes the outliers from the non-outliers using an adaptive threshold derived from the input XML. It is unrealistic to adopt a fixed threshold for all XML documents because the number of outliers varies according to different noise levels. Two adaptive strategies are examined, namely the *Top-k* and *Rate-of-Change* approaches. Finally, we remove redundancy of outliers that exist in overlapping neighbourhoods of varying degrees.

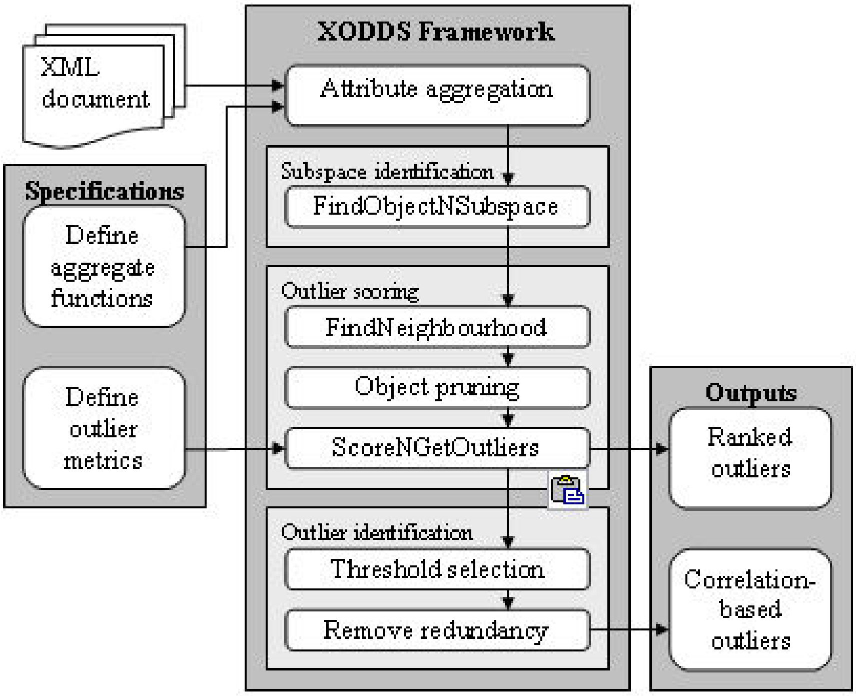


Figure 5.3: The XODDS outlier detection framework

5.3.2 Algorithms

The XODDS framework divides the outlier detection problem in XML into 4 systematic steps of attribute aggregation, subspace identification, outlier scoring, and outlier identification. In this section, we will describe the details of selected procedures in the framework.

Procedure *FindObjectNSubspace* identifies objects and subspaces in the XML document. If the elements or attributes are numerical, we discretize them into categorical values through binning them into equi-width intervals. After parsing the XML document into a DOM tree, the procedure performs a depth-first search on the tree [Line 1]. Each object node is collected into an array Object. If the node has the same sibling nodes, it is immediately identified as a subspace with its parent as the pivoting node [Line 5-6]. Otherwise, the object is stored into a list which is screened at Line 13-20. Following Definition 3 in Section 5.2.1, object nodes are checked if they belongs to the subspace S based on their nearest common neighbours and XPATHs. The function *isObjectNode* simply checks if the given node contains leaf nodes. Given that the XODDS algorithm is correlation-based, it relies on the correlation among 3 or more attributes to determine outlier-ness. The output of Procedure *FindObjectNSubspace* is a list of XML subspaces S.

```
Procedure FindObjectNSubspace
Input: XML document
Output: A list of XML subspaces, $S(v_p, l_s)$ defined over input tree $T$

Parse XML into DOM tree, $T$
$S \leftarrow \emptyset$, $Obj \leftarrow \emptyset$, $Object \leftarrow \emptyset$
For each $v_i$ of $T$ do
    If isObjectNode($v_i$) then
        If label(sibling($v_i$)) == label ($v_i$) then
            $S$(parent($v_i$), label($v_i$)) $\leftarrow v_i$
        Else
            $Obj \leftarrow v_i$, label($v_i$), XPATH($v_i$)
        Endif
        $Object(v_i) \leftarrow$ children($v_i$)
    Endif
Endfor
For each $v_i$, label($v_i$), XPATH($v_i$) in $Obj$ do
    For each remaining $v_j$, label($v_j$), XPATH($v_j$) in $Obj$ do
        If label($v_i$)== label($v_j$) AND XPATH($v_i$)== XPATH($v_j$) then
            $S$(NCA($v_i$, $v_j$), label($v_i$)) $\leftarrow v_i$
        Endif
    EndFor
EndFor
$S$(NCA($v_i$, $v_j$), label($v_j$)) $\leftarrow v_j$
```

Procedure *FindNeighbourhood* takes as input the list of subspaces identified and enumerates all the attribute combinations in each subspace, where each combination is a subset of the object's children leaf nodes [Line 1]. The support of each unique subset or neighborhood accumulates according to Definition 6 in Section 5.2.4 [Line 5-9]. The output of the table is a list of hash-tables for each attribute combinations.

```
Procedure FindNeighbourhood
Input: List of subspaces, S defined over input DOM tree T
Output: Set of support hashtables sup(V(k, $s$)) for each attribute

1.  Initialize support hashtables, V(k, $S(v_p$, label($v_s$))) based on projections of $v_s$
2.  For each subspace $S(v_p$, label($v_s$)) do
3.      For each node $v_i$ in $S(v_p$, label($v_s$)) do
4.          Create all subsets $s$ of degree k in $Object(v_i)$
5.          If V(k, $S(v_p$, label($v_s$)), $s$) exists then
6.              Increment count of sup(V(k, $S(v_p$, label($v_s$)), $s$)) by 1
7.          Else
8.              sup(V(k, $S(v_p$, label($v_s$)), $s$))  = 1
9.          EndIf
10.     EndFor
11. EndFor
```

Once the supports are computed, Procedure ScoreNGetOutliers compute the outlier scores based on the *x*O-measure or *x*Q-measure metric. For each target attribute of each projection in every subspace, the optimal thresholds based on the Rate-of-Change is computed [Line 7-14]. The calculate_score function returns the metric scores of v_a in all possible enumeration of correlated neighborhoods within $S(v_p, \text{label}(v_s))$. In order to reduce the cost of the extensive enumerations, a pruning step is introduced to calculate_score to filter combinations of target attributes and correlated neighbourhood (the equivalent of F_{11} discussed in next section) with supports larger than *minsup*. Line 7 sorts the attribute in ascending order while Line 8-10 computes the Rate-of-change from the sorted measures. Correlation-based outliers with the lowest ß outlier scores are output.

Procedure ScoreNGetOutliers

Input: List of supports of subsets. User option of outlier metrics. User input *minsup*
Output: Set of outlier measures for each attribute and subspace

```
1.  For each subspace $S(v_p, \text{label}(v_s))$ do
2.      For all k projection, $k \geq 3$ do
3.          For each node $v_a$ in $V(k, S(v_p, \text{label}(v_s)), s)$ do
4.              $A_k \leftarrow$ calculate_score($v_a$, $V(k, S(v_p, \text{label}(v_s)), s, minsup$)
5.          Endfor
6.      Endfor
7.      $B \leftarrow O_A$ sorted in ascending order of measure($A_k$)
8.      For each point $b_i$, $2 \leq i \leq |B_j|$ do
9.          Rate-of-change($b_i$) = $(b_i - b_{i-1}) / b_i$ // rate of change
10.     Endfor
11.     $ß \leftarrow$ i with max Rate-of-change($b_i$)
12.     For each $b_i$, $1 \leq j \leq ß$ do
13.         OUTPUT CA-outliers $\leftarrow b_i$
14.     Endfor
15. Endfor
```

5.4 Attribute Outlier Metrics

Interesting-ness measures are commonly used for ranking the usefulness and significance of the discovered co-relations or co-occurrences. They may therefore be appropriate for quantifying attribute outliers. In order to systematically compare the suitability of *x*O-measure and *x*Q-measure, as well as these interesting-ness metrics to the attribute outlier detection problem, we develop the key properties of an attribute outlier metric and evaluate the extent to which each metric satisfies these properties.

5.4.1 Interesting-ness Measures

Correlation and association-based measures for categorical data are commonly defined in terms of the frequency counts (or supports) in a 2 ×2 contingency table depicting the co-presence, co-absence and cross-presence of two variables. In detecting attribute outliers, we are interested in the correlation behaviour of a target attribute, denoted v_t, with remaining attributes of a correlated neighbourhood, denoted $N(v_t)$, as shown in Table 5.1.

Table 5.1: The 2×2 contigency table of a target attribute and its correlated neighbourhood

	$N(v_t)$	¬ $N(v_t)$	
v_t	F(v,N)	F	F_{1+}
¬ v_t	F_{-N}	F_{--}	F_{0+}
	F_{+N}	F_{+0}	N

Interesting-ness measures examined in this comparative study are mainly described in [HH99, TKS02]. Since attribute outlier-ness relates to negative correlations, we select metrics that measure negative correlations. We briefly describe these interesting-ness measures with respect to a correlated subspace of n objects in XML as follows:

5.4.1.1 Piatetsky-Shapiro Rule Interest

The Piatetsky-Shapiro (PS) Rule Interest [FPS99] calculates the difference between the observed and expected number of objects that contains both the target attribute v_t and the given correlated neighbourhood $N(v_t)$:

$$PS(v_t, N(v_t)) = \frac{F_{11}}{n} - \frac{F_{1+}F_{+1}}{n^2}$$

With values ranging from -0.25 to +0.25, the PS rule index measures both positive correlations (PS > 0) and negative correlations (PS < 0); PS = 0 indicates that v_t and $N(v_t)$ are statistically independent. Stronger attribute outliers are dictated by lower negative PS scores, meaning that the observed presence of v_t in $N(v_t)$ is lower than expected.

5.4.1.2 Interest factor

Instead of the difference, the Interest factor (I) computes the quotient of the observed and expected number of objects that contains v_t and $N(v_t)$ [TKS02]:

$$I(v_t, N(v_t)) = \frac{nF_{11}}{F_{1+}F_{+1}}$$

Values of I range from 0 to $+\infty$ where $I > 1$ indicates positive correlations and $I < 1$ implies that v_t negatively correlates with $N(v_t)$. The closer is the I values to 0, the stronger is the attribute outlier-ness (v_t, $N(v_t)$).

5.4.1.3 Jaccard coefficient

The Jaccard coefficient calculates the proportion of objects containing the target attribute v_t and the $N(v_t)$ over the cross-presence of v_t and $N(v_t)$ [JD98].

$$Jaccard(v_t, N(v_t)) = \frac{F_{11}}{F_{10} + F_{01} + F_{11}}$$

The Jaccard coefficient ranges between 0 to 1. Strong attribute outliers are represented by lower Jaccard coefficients close to 0 while positive correlations are depicted by values close to 1. Unlike Piatetsky-Shapiro and Interest, the Jaccard coefficient does not statistically distinguished between the positive and negative correlations.

5.4.1.4 Hmeasure

The Hmeasure is a multiplication of the individual effect of the cross-presence of v_t and $N(v_t)$ [HCH04]. Hmeasure ranges from 0 to 1 where a value close to 0 implies stronger positive correlations and that close to 1 implies high degree of negative correlations.

$$Hmeasure(v_t, N(v_t)) = \frac{F_{10}F_{01}}{F_{1+}F_{+1}}$$

A strong attribute outlier will therefore has Hmeasure close to 1, meaning that v_t has high degree of co-occurrences in other neighbourhoods, and $N(v_t)$ tend to co-occur with other attributes.

5.4.1.5 *Probability*

Probability is simply the probability that an object contains the target attribute v_t co-occurs with $N(v_t)$. The values range from 0 to 1 and strong attribute outliers are dictated by low degree of *Pr* close to 0.

$$\Pr(v_t, N(v_t)) = \frac{F_{11}}{n}$$

5.4.2 Properties of Attribute Outlier Metrics

5.4.2.1 *Metric properties*

Table 5.2: Example contingency tables for monotone properties.

M_2 indicates an attribute outlier, M_5 is a rare class, and M_6 depicts a rare attribute.

M_1	$N(v_t)$	$\neg N(v_t)$	
v_t	50	50	100
$\neg v_t$	50	50	100
	100	100	200

M_2	$N(v_t)$	$\neg N(v_t)$	
v_t	1	99	100
$\neg v_t$	99	1	100
	100	100	200

M_3	$N(v_t)$	$\neg N(v_t)$	
v_t	50	50	100
$\neg v_t$	450	50	500
	500	100	600

M_4	$N(v_t)$	$\neg N(v_t)$	
v_t	50	50	100
$\neg v_t$	50	450	500
	100	500	600

M_5	$N(v_t)$	$\neg N(v_t)$	
v_t	1	1	2
$\neg v_t$	1	99	100
	2	100	102

M_6	$N(v_t)$	$\neg N(v_t)$	
v_t	1	1	2
$\neg v_t$	99	1	100
	100	100	102

One of the first extensive works on the statistical properties of the correlation measures was presented by Piatetsky-Shapiro [Pia99]. Here, we first adapt 2 of the key properties proposed by Piatetsky-Shapiro to the attribute outlier problem. We use M to denote the matrix $\begin{bmatrix} F_{11} & F_{10} \\ F_{01} & F_{11} \end{bmatrix}$ and $o(M)$ to represent a metric function o applied to M.

Property 1 (Monotonically increases with F$_{11}$): Given that the overall supports of v_t as well as $N(v_t)$ do not change, $o(M)$ monotonically increases with the co-presence of v_t and $N(v_t)$. Property 1 ensures that $o(M)$ is statistically dependent on the significance of F$_{11}$. If the number of objects that indicate co-occurrences v_t and $N(v_t)$ decreases with respect to total number of objects that contains either v_t and $N(v_t)$, the greater is the likelihood that v_t is an attribute outlier with respect to $N(v_t)$.

For example in Table 5.2(a), *x*Q-measure(M_1)=50/100=0.5 while M_2 with reduced F_{11} has a corresponding lower *x*Q-measure(M_2)=1/100=0.01. Therefore, *x*Q-measure satisfies Property 1.

Property 2 (Monotonically decreases with F_{1+} or F_{+1}): *o*(*M*) monotonically decreases when the supports of v_t or $N(v_t)$ increases, given that the co-presence does not changed. Property 2 indicates that $o(M)$ is lower (and therefore v_t exhibits higher degree of outlier-ness in $N(v_t)$) if a greater coverage of v_t or $N(v_t)$ is required to achieve the same level of co-presence. *x*Q-measure also satisfies Property 2; the *x*Q-measure of M_2 is 0.01 is comparatively lower than *x*Q-measure(M_3)= 50/500=0.1.

Property 3 (Null invariance): *o*(*M*+C)= o(*M*) where C = $\begin{bmatrix} 0 & 0 \\ 0 & k \end{bmatrix}$ and k > 0. Sparse data has high degree of F_{00}. Property 3 ensures that the attribute outlier metric for XML is invariant to F_{00}, the co-absence of v_t and $N(v_t)$. This property is often known as the Null invariance property [TKS02, HCH04]. For instance, M_1 and M_4 only differs in the magnitude of F_{00} and *x*Q-measure(M_1)=*x*Q-measure(M_4)=50/100=0.5.

Property 4 (Rare class): *o*(*M*) differentiates rare classes from attribute outliers. Often, rare classes (or rare objects) are mistaken as attribute outliers. For example, consider the Branch_Address object defined by the elements - <Country:Micronesia>, <State:Ponape>, and <City:Palikir>. None of the 3 elements are attribute outliers because Micronesia consistently co-occur with Ponape as well as Palikir and vice versa. M_5 in Table 5.2 depicts the contingency table of a rare object which must be differentiated from the real attribute outlier described in $M_{2.}$ *x*Q-measure satisfy Property 4; *x*Q-measure(M_5)=1/2=0.5 is comparatively higher than *x*Q-measure(M_2).

Property 5 (Rare attribute): *o*(*M*) differentiates rare attributes from attribute outliers. Property 5 states that the metric differentiates rare attributes such as the <Country:Zech> element in Figure 1(b) from the attribute outliers. A metric that satisfies Property 5 is not necessarily a good attribute outlier metric, but rather it depends on the nature of the input data and the user requirements. Rare attributes may be the result of errors, so data cleaning processes, which require that they are isolated together with attribute outliers, may specifically select metrics that do not satisfy this property. For example, *x*Q-measure(M_6)=1/100=0.01 is equivalently low as *x*Q-measure(M_2); *x*Q-measure does not differentiate between the attribute outliers and rare attributes.

Property 6 (Downward closure: *o*(*M*) is downward closure with respect to *N*(v_t). Property 6 is the basis of support-based pruning. Let $N'(v_t) \subseteq N(v_t)$ and o'(M) be the value of the measure on v_t and $N'(v_t)$, a metric satisfies *downward closure* if $o'(M) \geq o(M)$. The property ensures that if v_t=<Country:USA> is an attribute outlier in the neighborhood $N(v_t)$={<State:S.Moravi>,<City:Opava>,<Continent:Europe>}, then v_t is also an attribute outlier with respect to {<State:S.Moravi>,<Continent:Europe>}.

5.4.2.2 Evaluation of Attribute Outlier Metrics

As shown in Table 5.3, none of 8 metrics investigated satisfies all 6 properties, we note that some properties are more important than others. Property 1 and 2 statistically justify the correctness of the metrics. Property 3 ensures that the metric is least affected by sparse data. Notice that both Piatetsky-Shapiro and Interest do not satisfy Property 3. This means that the effectiveness of these two metrics are limited by the sparseness of the XML data.

Property 4 distinguishes outlier class from attribute outlier and therefore is a critical property for accurate attribute outlier detection, especially if the metric is used for error identification. Notice that the Probability metric does not satisfy most properties. Given that probability merely measures the rarity of a target attribute and its neighborhood, it is obviously an inappropriate measurement for attribute outliers and the experiment in the next section justifies this observation.

Property 5 and 6 are optional features that depend on the nature of the application and the efficiency requirements. Notice that *x*O-measure and *x*Q-measure satisfy all properties except for Property 5; they do not differentiate rare attributes and attribute outliers. Both *x*O-measure and Jaccard do not enable support-based pruning.

Table 5.3. Properties of attribute outlier metrics

Metrics	P1	P2	P3	P4	P5	P6
*x*O-measure	Yes	Yes*	Yes	Yes	No	No
*x*Q-measure	Yes	Yes*	Yes	Yes	No	Yes
Piatetsky-Shapiro (PS)	Yes	Yes	No	Yes	Yes	No
Interest (I)	Yes	Yes	No	Yes	Yes	Yes
Jaccard (ζ)	Yes	Yes	Yes	Yes	No	No
H-measure=(1-H-measure)	No	No	Yes	Yes	Yes	No
H-measure'	Yes	Yes	Yes	Yes	Yes	No
Probability (Pr)	Yes	No	Yes	No	No	Yes

* Only if the property applies to increase of the supports of $N(v_t)$

5.5 Performance Evaluation

We evaluate the performance of the XODDS algorithm on both synthetic Bank Account and real-world protein data sets. Experiments were performed on an Intel Core 2 dual, 2 GHz computer with 1GB of main memory, and running Mac OS. Programs are written using Java. The following aspects are investigated:

(1) Compare the accuracy of *x*O-measure, *x*Q-measure and other interesting-ness metrics using Top-k and Rate-of-Change selection methods.

(2) Evaluate the accuracy of various attribute outlier metrics in data sets with varying noise levels.

(3) Compare the accuracy of XODDS with the relational approach which uses χ^2 to derive the correlated subspaces in relational tables.

(4) Evaluate the performance of XODDS at higher level of abstractions through the use of aggregate attributes. In particular, we are looking for "hidden" outliers which could be missed if not for the use of aggregate attributes.

(5) Evaluate the running time of attribute outlier metrics at varying data size, with and without pruning.

(6) Evaluate the performance of XODDS on UniProtKB/TrEMBL database to detect discrepancies in annotations. The results are manually annotated.

5.5.1 Bank Account Data Set

We extracted from the financial data set available at the web site http://lisp.vse.cz/pkdd99/Challenge/berka.htm, a Bank Account data set (denoted Bank) containing 4,500 accounts, 207,989 transactional records, 670 loan records, and 711 payment orders. The data set, originally in relational tables, is converted to the XML schema as shown in Figure 5.4. To prevent the implicit noise from interfering with the evaluation process, we removed 4,884 transactional records which potentially contain outliers, leaving 203,105 transactions.

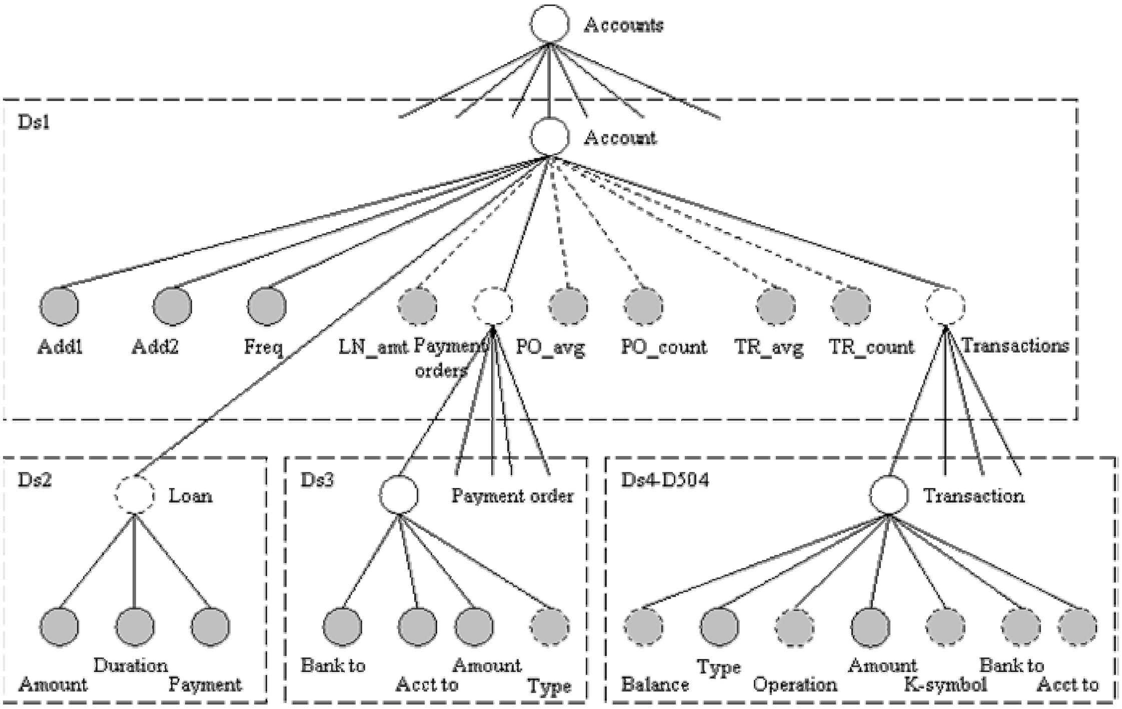


Figure 5.4: XML tree of Bank Account data partitioned to 504 subspaces (Ds1-504)

Aggregate attributes specified for this data set are Loan_amt, PO_count, PO_avg, TR_avg, and TR_count. Optional attributes are in dotted nodes.

5.5.1.1 Top-K vs Rate-of-Change

We compare the performance of various metrics on the Bank Account data set with 2% attribute outliers (as well as rare attributes) inserted. Figures 5.5 and 5.6 show that *x*O-measure and *x*Q-measure which are metrics specifically designed for attribute outlier detection, generally outperform the other metrics using either Top-k or Rate-of-Change (ROC) threshold selection strategy.

In Figure 5.5, the F-scores for *x*O-measure and *x*Q-measure converges up 80% when outliers are distinguished using the ROC thresholds are calculated from the top 15% of the objects (k) with the lowest metric scores. Figure 5.6 shows that the performance of the same metrics without ROC, at corresponding F-scores of up to 66%. Also noticed that at k=1%, the top-k approach achieves much higher F-scores than ROC across all metrics. This is expected because the input XML document contains 2% noise; selecting a "tighter bound" within the top 1% outliers merely serves to increase the number of FNs (false negatives).

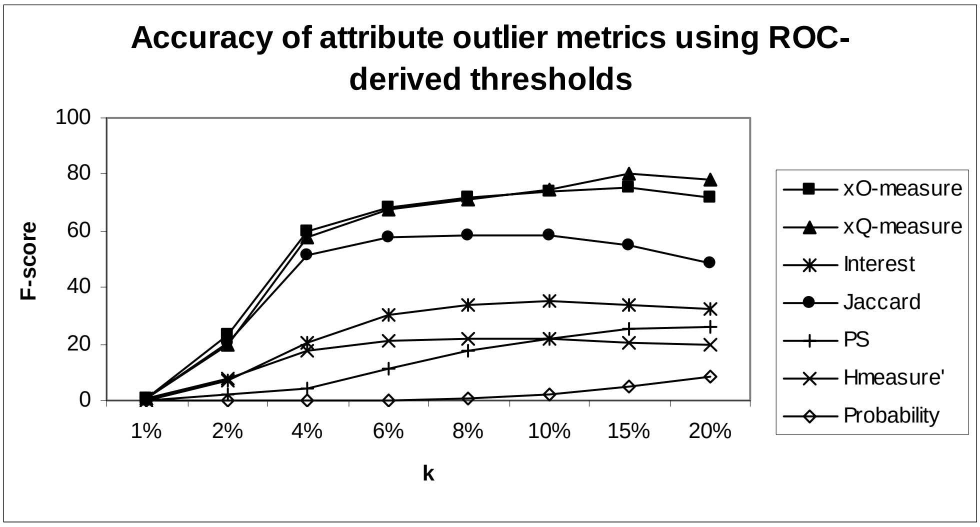


Figure 5.5: Performance of XODDS of various metrics using ROC-derived thresholds

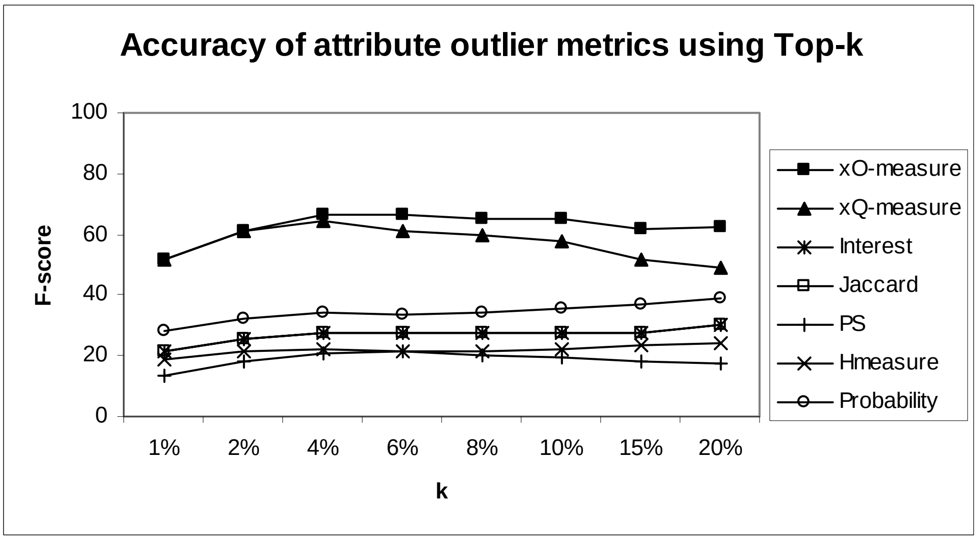


Figure 5.6: Performance of XODDS of various outlier metrics using Top-k

From a realistic perspective, we inserted attribute outliers as well as rare attributes into the input XML document because both types of patterns entail attention. Potentially fraudulent patterns such as the $1,000 transactional amount in Figure 5.1(c) are present as rare attributes, whereas abnormally low (or high) transactional averages compared to other accounts of the same country are seen as attribute outliers. Hence, metrics that do not differentiate between them - Property 5 (*x*O-measure, *x*Q-measure and Jaccard) generally achieves higher F-scores. One other reason for the poor performance of Interest and PS metrics is their dependence on the vast co-absence of XML data; they do not satisfy the null invariance property.

5.5.1.2 Varying Noise

Figure 5.7 compares the F-scores of various metrics at varying noise levels. Outlier thresholds are adaptively selected by XODDS using ROC. *x*O-measure and *x*Q-measure perform consistently better than the other metrics, even as the noise level increases. F-scores range between 72-81% for *x*O-measure and between 75%-80% for *x*Q-measure.

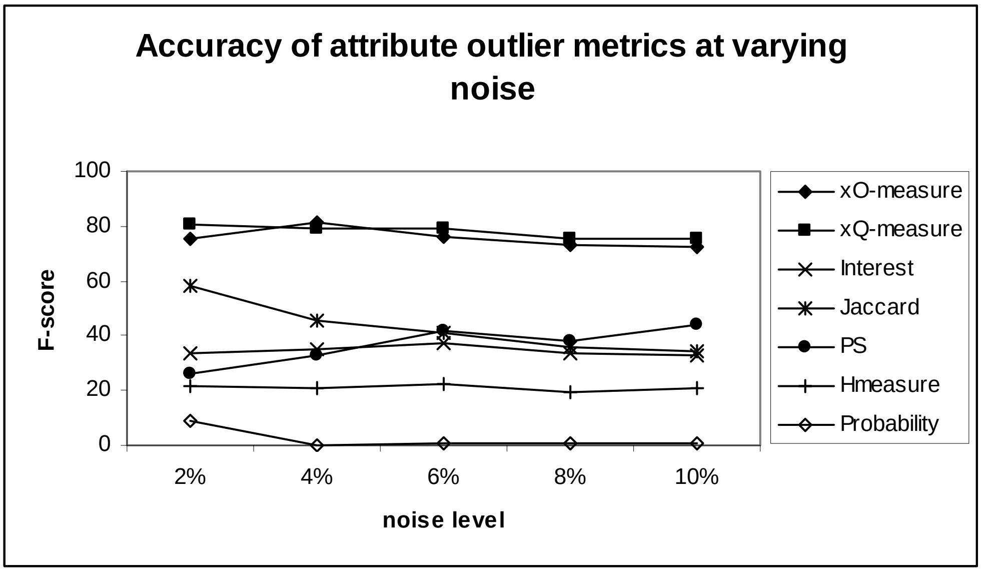


Figure 5.7: Performance of XODDS at varying noise levels

5.5.1.3 XODDS vs Relational

XODDS utilizes the inherent hierarchical structure of XML to derive meaningful subspaces for outlier detection. Similarly, given a relational table, dimensions or columns can be divided into subspaces through *attribute clustering*. In order to compare the accuracy of detecting outliers using these two approaches, we relationalizes the Bank Account data set (denoted RBank), and cluster the attributes based on Chi-square χ^2.

The χ^2 test is typically used to determine correlations between dimensions containing categorical data [Eve77]. For non-parametric data or continuous data, rank-order correlation calculations such *Spearman-Rho* and *Kendall-tau* or the *Pearson* test can be used instead. The χ^2 test of independence is based on the difference of the observed frequencies with the corresponding expected frequencies. If χ^2 = 0, the two attribute vectors are statistically independent. The χ^2 is computed by:

$$\chi^2 = \sum \frac{\left(f_{obs} - f_{\exp}\right)^2}{f_{\exp}}$$

where f_{exp} refers to the expected frequency of an attribute pair in the contingency table and f_{obs} is the observed frequency.

Table 5.4: Attribute subspaces derived in RBank using χ^2

Subspace	Attributes
1	Freq, Add1, Add2, Transaction/Acct to, Transaction/Bank to, Payment order/Acct to
2	Loan/Amount, Loan/Duration, Loan/Payment
3	Payment order/Bank to, Payment order/Amount, Payment order/Type
4	Transaction/Type, Transaction/Operation, Transaction/Amount, Transaction/Balance, Transaction/K-symbol

Table 5.4 shows the groups of attributes derived from the relational table using χ^2. It makes sense that the region(Add1) or district(Add2) where the customer open his bank account is likely dependent on the banks he often transacts to, it is not surprising to find that Transaction/Bank to, Transaction/Acct to, and Payment order/Acct to are correlated to the Bank branch addresses (Add1 and Add2) in Subspace 1. We apply the same outlier scoring and selection algorithms to the relational subspaces. Figure 5.8 shows the performance of XODDS compared with the relational approach at varying noise levels. As discussed in Section 5.2, the poor performance of the latter approach may be due to the increased redundancy when an object is replicated across multiple tuples when "flattened", and thus affecting the distribution of each dimension in the converted relational table.

XODDS consistently performs better compared to the relational approach with F-scores of between 63%-86%. As the noise level increases, the correlation between the attributes decreases, thus affecting the accuracy of the outlier detection. Overall, the accuracy using *x*Q-measure in XODDS is slightly higher than with *x*O-measure, particularly when the level of noise increases. In fact, even on the Bank Account data set with 10% noise, the F-score achieved using XODDS with *x*Q-measure is 70%.

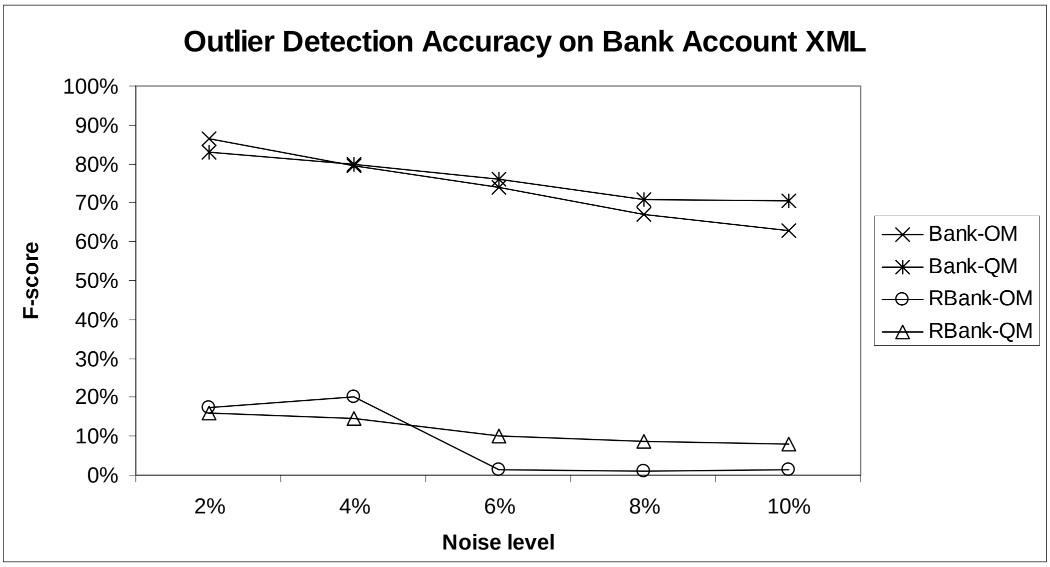


Figure 5.8: Performance of XODDS compared to the relational approach

5.5.1.4 Aggregate attributes

The purpose of aggregate attributes is to summarize nested XML structures so that detection of attribute outliers at higher level of abstraction is possible. We introduced 5 aggregate attributes to the Bank Account data set: *TR_count* and *TR_avg* are the number of transactions and the average amount of transactions of an account. *PO_count* and *PO_avg* are the number of payment orders and the average amount of the payment orders. *LN_amt* is the loan amount. Since each account is restricted to one loan, the loan amount is merely "promoted" to the account level.

The aggregate attributes are compared across all 4,500 accounts in the Bank Account data set; the outliers identified are shown in Figure 5.9. We are interested to determine any inherent outliers in the raw data set. We anticipated that removing 4,884 transactional records with possible outliers in the transaction subspace may not necessarily "clean up" the account subspace. This is justified by applying XODD on the clean data set which does not contain any inserted outliers. Some of the interesting inherent outliers which are uncovered are:

1. An account which has less than 10 transactional records.
2. Loan amount of less than $1,000 issued to 2 accounts.
3. Loan amount of more than $100,000 issued to 1 account.

4. 18 accounts in Hl.m. Praha city of Czech Republic have transactional averages less than $100. These accounts are highlighted because the typical transactional averages of accounts opened in Hl.m. Praha is usually in magnitude more than $100.

In general, the number of aggregate attribute outliers does not increase significantly as noise level increases, even for the averaging attributes. Inserting fraudulently high transactional amount to the transaction object may increase the outliers at the account level (e.g. transactional average increases) but it may also average out the outlier behavior.

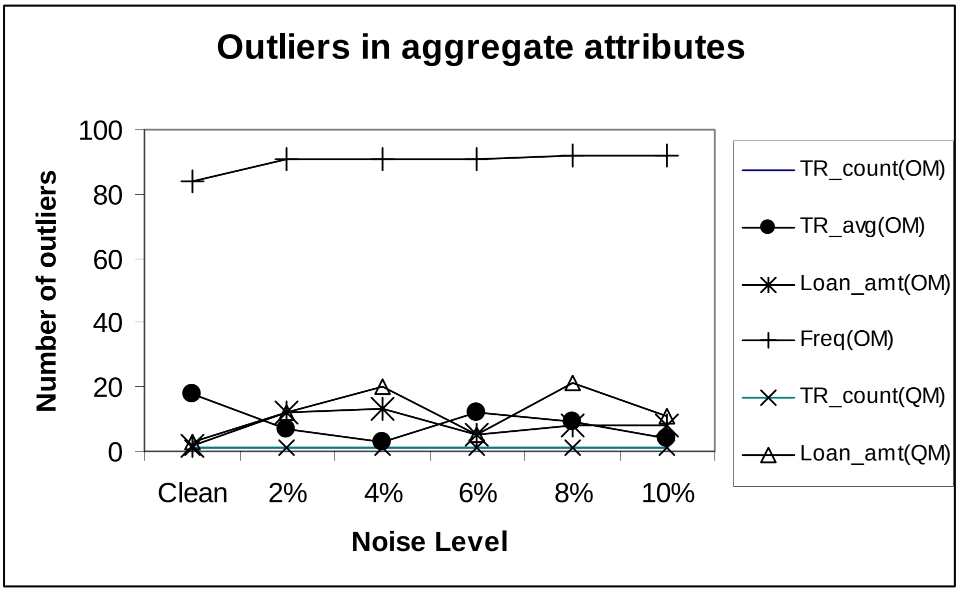


Figure 5.9: Number of aggregate outliers in the account subspace across varying noise

5.5.1.5 Running time at varying data size

We evaluate the running time of XODDS across varying data size. As shown in Figure 5.10, XODDS increases exponentially. At 9,000 account nodes, XODDS requires 38 minutes. Moreover, pruning of attribute outliers that have supports greater than *minsup* significantly reduce the execution time (details in Section 5.3.2).

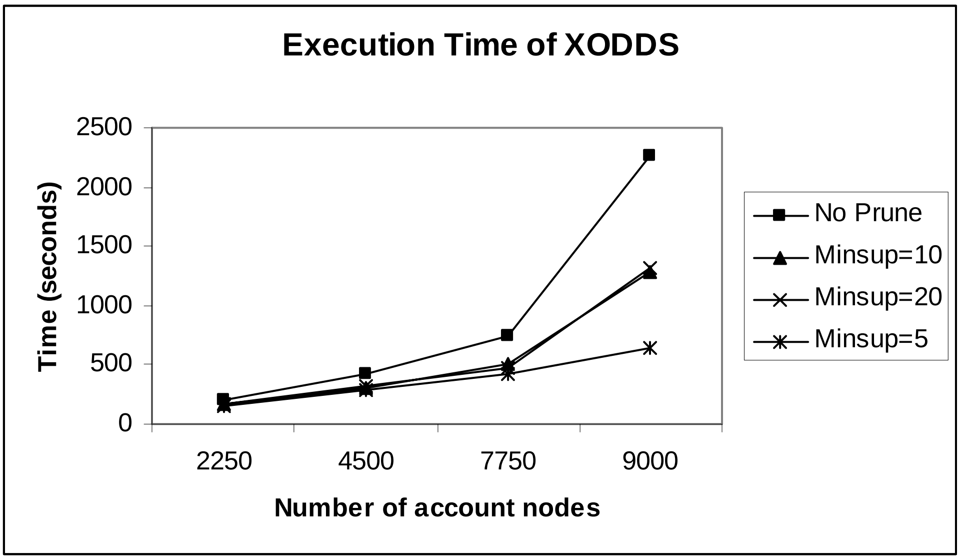


Figure 5.10: Running time of XODDS at varying data size

5.5.2 UniProt Data Set

The second part of the experiments evaluate the performance of XODDS in detecting annotation errors in a the UniProt/TrEMBL data set, and shows that XODDS is achieve better accuracy compare to the ODDS method proposed in Chapter 4.

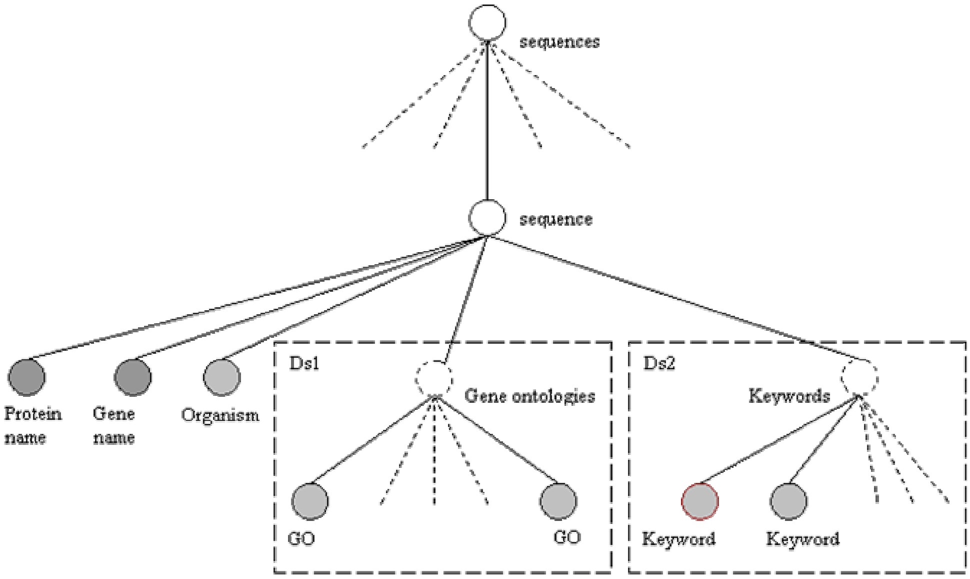


Figure 5.11: Simplified UniProt XML

The original UniProt XML schema is extensive. For purpose of this study, we simplify the UniProt XML (Figure 5.11) and only focused on selected subspaces of the **Gene Ontologies** and the **Keywords**. Each protein object in UniProt/TrEMBL is annotated with a set of Gene ontology nodes corresponding to the Gene ontology controlled vocabulary of proteins' properties. Similarly, a protein contains a number of keyword subject references. Table 5.5 shows the detected outliers. The k-neighbourhoods indicate the size of the neighbourhoods of the outliers.

Table 5.5: Outliers detected from the UniProt/TrEMBL Gene Ontologies and Keywords annotations

k-Neighborhoods	GO (OM)	GO(QM)	KW(OM)	KW(QM)
3	46	184	5	40
4	91	491	6	53
5	54	251	49	30
6	60	85	53	25
7	10	13	75	11
8	904	473	65	2
9	532	99	215	228
10	9	9	43	44
11	3712	9595	4	4
12	1621	4252	1	1
13	524	1391	2	2
14	117	317	-	-
15	16	45	-	-
16	1	3	-	-
17	3	5731	-	-
18	1208	765	-	-
19	75	50	-	-
Total	**8983**	**23754**	**418**	**440**

A biologist through manual verification verifies the detected attribute outliers. Table 5.6 shows the accuracy of the Gene ontology outliers detected using ODDS (with O-measure). True positive TP indicates an uncommon association of the target attribute with the other attributes in the projected relation. False positive FP indicates that no peculiarity is found in the correlation behaviour of the target attribute. Indeterminable means that further investigation is required.

The manual verification step largely depends on the knowledge level of the biologist and his decisive-ness. Table 5.6 shows that a large percentage 43% of the outliers require further investigation because the biologist lacks the domain knowledge to justify if the annotation is erroneous or it is only exceptional. Only 3% out of those which can be annotated are false positives. The remaining 97% of the gene ontology outliers are confirmed erroneous annotations. This is a significant improvement over the 19%-55% positive predictive rate (PPV) achieved by ODDS (details in Chapter 4, Section 4.5.2).

Table 5.6: Annotation results of outliers detected from the UniProt/TrEMBL Gene ontologies

k-Neighborhoods	TP	FP	Indeterminable
3	9	12	25
4	28	7	56
5	23	3	28
6	35	0	25
7	6	0	4
8	283	48	573
9	189	39	304
10	4	1	4
11	2347	5	1360
12	1167	0	454
13	419	0	105
14	102	0	15
15	15	0	1
16	1	0	0
17	1	0	2
18	354	33	821
19	21	1	53
Total	**5004**	**149**	**3830**

To mention a few interesting examples, a Mitochondrion protein Q35127 which occurs outside the nucleus is annotated to be involved in "DNA repair" - a process which occurs only inside the nucleus, a Chloroplast protein Q5UVG7 which is a plant cell organelle, is involved in "defense response to pathogen" and another Mitochondrion protein Q9B8V5 is annotated to contain a "nucleus". Some of these erroneous annotations are corrected in the latest UniProtKB/TrEMBL release.

5.6 Concluding Section

In this chapter, we have presented a novel framework called XODDS that utilizes the correlations between attributes to identify, attribute outliers. We introduced in XODDS, two new concepts of correlated subspace and aggregate attributes which are derived from the hierarchical structural of XML data models. We also develop two attribute outlier metrics - *x*O-measure and *x*Q-measure. Through evaluation with synthetic and real-world data, we have shown that XODDS can achieve F-score of up to 80% and it can determine up to 97% erroneous annotations on a real-world protein data set.

Chapter 6: Duplicate Detection from Association Mining

Nothing in life is to be feared, it is only to be understood.

Marie Curie
Chemist and Physicist (1867 - 1934)

Duplicate detection is an important data cleaning problem; duplicates cause over-representation of patterns and affect the accuracy of the data mining results. Existing duplicate detection methods consider redundancy as a Boolean operation – if two records are sufficiently similar, they are identified as duplicates and are merged. However, for many real-world databases, multiple facets of duplication exist and the merging step depends on the types of redundancy detected. Not all duplicates can be trivially combined. One such example is the biological data. In Chapter 3 (Section 3.2.2.2), we discuss the limitations of using a simplified rule to detect varying types of duplicates.

This chapter explores a learning approach towards detecting multiple types of duplicate relations. Leveraging on the correlations between attributes, duplicate rules for different types of duplicate relations are induced from a known set of duplicates using association mining. The method is used to identify 5 types of duplicates in biological data – duplicates, structural isoforms, cross-species duplicates, sequence fragments, and cross-annotation variants.

Evaluation of the proposed approach on a protein data set shows that the duplicate rules are capable of identifying up to 97.3% of the varying types of duplicates. Slight improvement is achieved over other classifiers, but the approach has practical advantage of requiring only the positive training set of duplicates. Other classifiers are highly dependent on the completeness of the comparatively larger negative training set.

6.1 Introduction

Extensive diversity in data formats, schemas, nomenclatures as well as in geographical distribution introduces in high level of information redundancy among the biological databases. The same sequence may have inconsistent, overlapping, or partial information in heterogeneous representations in heterogeneous data sources. The various causes of redundancy in biological data are described in Chapter 3 (Section 3.1.3).

In a comprehensive study of the structural and functional annotations of scorpion toxins [SGT$^+$02], we observed through collecting and merging the scorpion toxin records from 6 source databases: 143 out of 211 (68% redundancy) scorpion toxin proteins are replicated across 2 to 5 data sources; 13 are found in 5 database sources (Figure 6.1).

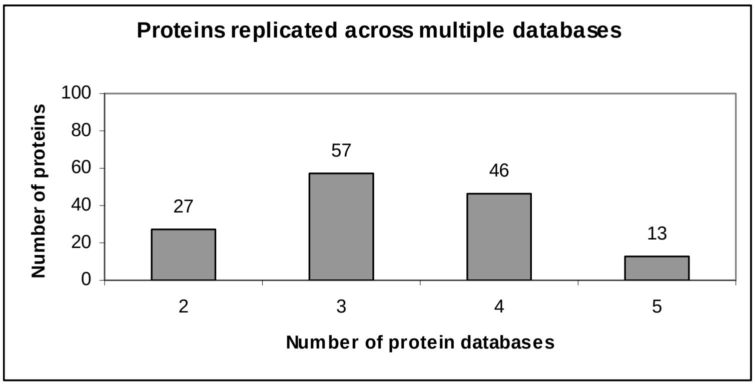


Figure 6.1: Extent of replication of scorpion toxin proteins across multiple databases

For merely 211 records, it is possible to delete or merge-join duplicates manually. But this is not viable for large-scale functional and structural studies of other organisms such as fugu (more than 2,000 protein records in GenBank as of Dec 2006) or mouse (more than 180,000 protein records in GenBank as of Dec 2006), let alone human (more than 320,000 protein records in GenBank as of Dec 2006).

Typically, cleaning of biological data is carried out in proprietary or ad-hoc manner, sometimes even manual. Systematic processes for biological data cleaning are lacking. Rather, specific procedures are designed for cleaning certain datasets. For example, in [Tha99], stringent selection criteria are used to select complete and unique records of Homo sapiens splice sites from EMBL database. Requirements for complete coding region, genuine human nuclear DNA, non-pseudogene, absence of alternative gene products, among others, reduced the initial 4,300 raw records to 400 records. Retaining only one sequence among any group of sequences with more than 80% identity reduced the dataset further to 310 records. Such approaches of eliminating all incomplete records (sequences) can result in loss of information from cases of partially determined or partially annotated yet non-redundant sequences.

6.1.1 Motivating Example

Existing duplicate detection methods focus primarily on well-defined records such as customer contact information. On the other hand, biological data records have more than one facet of duplication. For example, the known duplicate relations of annotated protein sequences in the NCBI entrez searchable protein databases (GenPept, Swiss-Prot, PDB, PIR, among others) are described in Table 6.1.

Table 6.1: Multiple types of duplicates that exist in the protein databases

Duplicate type	Description
1. Duplicates	Identical protein sequences with different annotations (one may be a structure record) due to: • Sequences submitted by different annotators • Sequences submitted more than once to same database • Sequences submitted to different databases
2. Structural isoforms	Same protein sequence but records have different orientations or conformations due to: • Difference in partial organisation of proteins (Chain A and B) • Structural modeling of the same protein by different methods or by different researchers • Cleavage resulting in different foldings • Protein complexes • Sequence from structural inhibition study
3. Cross-species duplicates	Identical protein sequences exist in different genus/species (not subspecies)
4. Sequence fragments	One sequence is a segment of a more complete sequence due to: • Partially determined sequence fragment • Precursor/mature protein pairs (both are complete sequences)
5. Cross-annotation variants	Highly similar sequences but not identical sequences with different features in annotation due to: • Sequence variants with different functions • Synthetic sequences for the study of functional residues • Different annotations given by separate annotators

Duplicates refer to proteins which are recorded in more than one database entries due to different data sources, varying views of the proteins (PDB protein structures versus Swiss-Prot protein annotations), or repeated submissions of the sequence by the same or different annotators. Usually, duplicates contain partial information of the same protein sequence and should be merged into a single entity. *Structural isoforms* are database records describing different structural conformations or orientations of the same protein. *Cross-species duplicates* are identical protein sequences belonging to different species. *Sequence fragments* contain partial information of the complete sequence, and are typically merged into the latter. Traditionally, incomplete sequences are eliminated during data cleaning processes. This approach, however, may result in loss of important information. *Cross-annotation variants* are highly similar sequences with slight difference in sequence features. Biologically, cross-annotation variants are particularly useful in identifying specific residues that are critical in determining the protein's functional properties.

Generally, the actions taken upon detecting the varying types of duplicates depend on the objectives of the users as well as on the type of the duplicate relations. For example, duplicates may be directly merged to form complete sequences while cross-annotation variants will need to be inspected separately by the expert annotators. Merging structural isoforms depends on the nature of the analysis; whether the analysis is based on the protein structures or the primary sequences.

The same multiplicity of duplicate relations also exists in non-biological domain. For example, two patient records that differ only in the blood group cannot be directly combined. Possible contamination during blood tests may result in labelling of incorrect blood groups. This in turn may result in serious loss of lives [KCD99]. Such duplicate records must be highlighted so that additional blood tests are conducted.

Specific contributions from this work include:

1. We introduce the notion of multiple types of duplicate relations, as opposed to traditional concept that redundancy is a boolean property.

2. We propose a correlation-based method for learning the duplicate rules of varying types of duplicate relations. Different duplicate rules are induced from a known set of duplicates using association mining.

The rest of the chapter is organised as follows: In section 6.2, we briefly describe association mining – the basis of our duplicate learning approach. Section 6.3 details the materials and methods. In section 6.4, we discuss results of our experiments and we conclude in section 6.5.

6.2 Background

Learning techniques that rely on supervised classifiers for duplicate detection are not new. [SB02] proposed an iterative de-duplication system that actively learns using Decision Tree C4.5, Support Vector Machine (SVM), and Naïve Bayes as the classifiers. [EVE02] utilizes probabilistic, induction and clustering based decision models to classify the records into two classes – duplicates and non-duplicates. Unlike these methods which require a negative training set of non-duplicates, the input to the association mining are pairs of duplicates. Through association-based classifier approach, each type of duplicate relations is characterized by both the similarities of the attributes (which we call *matching criteria*) as well as the correlation patterns among the attributes.

In Section 6.4 of this chapter, we will show that association-based classifier outperforms other classifiers.

6.2.1 Association mining

Association mining or induction is commonly used in market basket analysis to find items frequently bought together by shoppers. The first algorithm for mining frequent item sets is *Apriori* [AIS93]. Association rules are induced from items that are most frequently occurred together, known as the *frequent item set*. For example, a rule of the form “Buy(A) ^ Buy(B) → Buy(C)” indicates that a customer who buys item A and item B buys C, with the interestingness of this rule measured from the support and confidence of the rule. The support is the percentage of transactions in the input database that contain A, B and C. The confidence is the percentage of transactions that contain A and B (the antecedent) also containing C (the consequent).

Association rule mining has been applied to other data cleaning problems [MML01, LLH04] but not in duplicate detection. As such, this is the first application of association mining methods on duplicate detection problem. To the best of our knowledge, this is also the first comprehensive work that addresses redundancy in biological data.

6.3 Materials and Methods

This section details the duplicate detection framework.

6.3.1 Duplicate Detection Framework

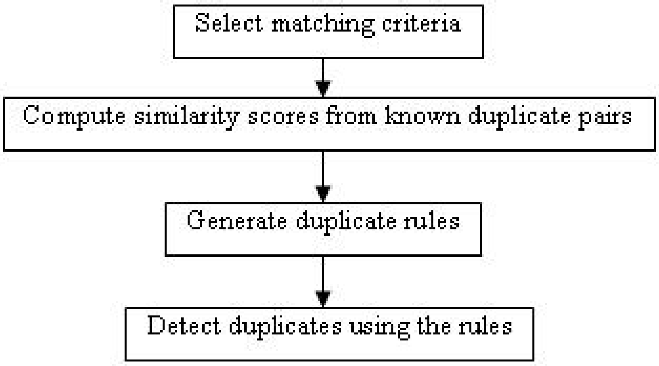
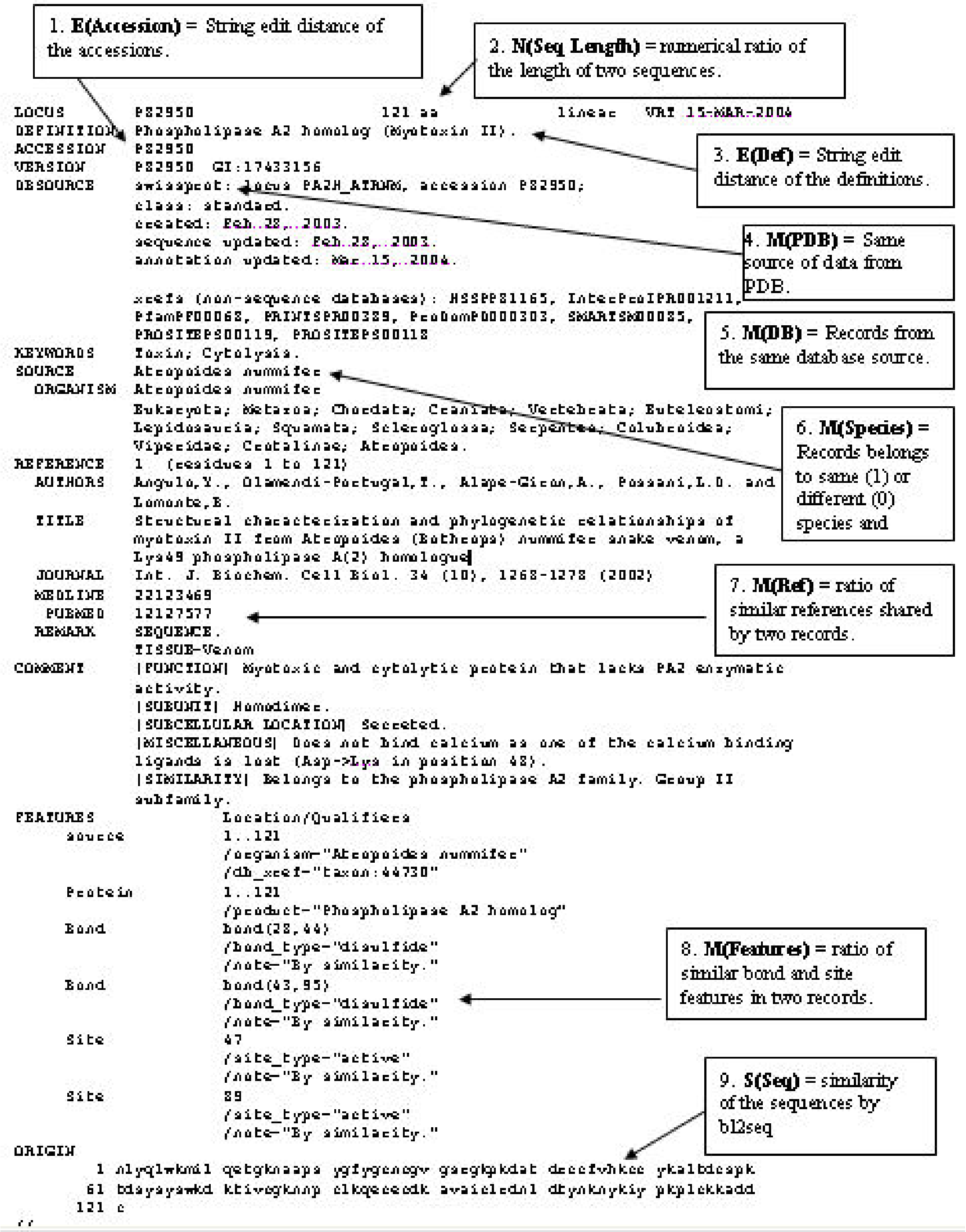

Select matching criteria
Compute similarity scores from known duplicate pairs
Generate duplicate rules
Detect duplicates using the rules

Figure 6.2: Duplicate detection framework

Figure 6.2 depicts the association-based duplicate detection framework. First, matching criteria for comparing record pairs are selected from the input data set. Selected attributes based on these matching criteria of each duplicate record pairs are measured using varying similarity functions, depending on the data types of the attributes. The similarity values for each pair of records in the training data are discretized. Duplicate rules mined from subsets of attributes and their similarity measures, each describing a type of duplicate relation are used to detect duplicates in biological datasets.

6.3.2 Matching Criteria

Protein records from Entrez are comparable across 9 matching criteria as shown in Figure 6.3. A protein record contains 3 main types of data fields: (1) Protein and DNA primary sequences, (2) categorical fields, and (3) free-text strings, each requiring different similarity functions to measure the degree of similarity of two corresponding fields.



1. **E(Accession)** = String edit distance of the accessions.

2. **N(Seq Length)** = numerical ratio of the length of two sequences.

3. **E(Def)** = String edit distance of the definitions.

4. **M(PDB)** = Same source of data from PDB.

5. **M(DB)** = Records from the same database source.

6. **M(Species)** = Records belongs to same (1) or different (0) species and

7. **M(Ref)** = ratio of similar references shared by two records.

8. **M(Features)** = ratio of similar bond and site features in two records.

9. **S(Seq)** = similarity of the sequences by bl2seq

```
LOCUS       P82950                  121 aa            linear   VRT 15-MAR-2004
DEFINITION  Phospholipase A2 homolog (Myotoxin II).
ACCESSION   P82950
VERSION     P82950  GI:17433156
DBSOURCE    swissprot: locus PA2H_ATRNM, accession P82950;
            class: standard.
            created: Feb 28, 2003.
            sequence updated: Feb 28, 2003.
            annotation updated: Mar 15, 2004.

            xrefs (non-sequence databases): HSSPP81165, InterProIPR001211,
            PfamPF00068, PRINTSPR00389, ProDomPD000303, SMARTSM00085,
            PROSITEPS00119, PROSITEPS00118
KEYWORDS    Toxin; Cytolysis.
SOURCE      Atropoides nummifer
  ORGANISM  Atropoides nummifer
            Eukaryota; Metazoa; Chordata; Craniata; Vertebrata; Euteleostomi;
            Lepidosauria; Squamata; Scleroglossa; Serpentes; Colubroidea;
            Viperidae; Crotalinae; Atropoides.
REFERENCE   1  (residues 1 to 121)
  AUTHORS   Angulo,Y., Olamendi-Portugal,T., Alape-Giron,A., Possani,L.D. and
            Lomonte,B.
  TITLE     Structural characterization and phylogenetic relationships of
            myotoxin II from Atropoides (Bothrops) nummifer snake venom, a
            Lys49 phospholipase A(2) homologue
  JOURNAL   Int. J. Biochem. Cell Biol. 34 (10), 1268-1278 (2002)
  MEDLINE   22123469
   PUBMED   12127577
  REMARK    SEQUENCE.
            TISSUE=Venom
COMMENT     |FUNCTION| Myotoxic and cytolytic protein that lacks PA2 enzymatic
            activity.
            |SUBUNIT| Homodimer.
            |SUBCELLULAR LOCATION| Secreted.
            |MISCELLANEOUS| Does not bind calcium as one of the calcium binding
            ligands is lost (Asp->Lys in position 48).
            |SIMILARITY| Belongs to the phospholipase A2 family. Group II
            subfamily.
FEATURES             Location/Qualifiers
     source          1..121
                     /organism="Atropoides nummifer"
                     /db_xref="taxon:44730"
     Protein         1..121
                     /product="Phospholipase A2 homolog"
     Bond            bond(28,44)
                     /bond_type="disulfide"
                     /note="By similarity."
     Bond            bond(43,95)
                     /bond_type="disulfide"
                     /note="By similarity."
     Site            47
                     /site_type="active"
                     /note="By similarity."
     Site            89
                     /site_type="active"
                     /note="By similarity."
ORIGIN
        1 slyqlwkmil qetgknaaps ygfygcncgv grrgkpkdat drccfvhkcc ykaltdcspk
       61 tdsysyswkd ktivcgknnp clkqececdk avaiclrdnl dtynknykiy pkplckkadd
      121 c
//
```

Figure 6.3: Matching criteria of an Entrez protein record

Protein or DNA sequences are matched using their percentage identity scores computed from BLAST 2 sequences (bl2seq) algorithm [TM99]. Bl2seq utilize the gapped BLAST algorithm [AMS$^+$97] to align and compare pairs of DNA-DNA or protein-protein sequences, and the percentage identity scores reflect the degree of similarity of the two sequences. We denote the sequence similarity function as *S*.

Categorical fields contain values belonging to a fixed value-set. For example, the organism fields in Entrez records are derived from the standardized taxonomy of the organisms. If two categorical fields have the same values, they are given a similarity score of 1; otherwise 0. We denote this Boolean match as similarity function *M*.

The third type of data fields are the free-text strings. The most common method for comparing string is the edit distance or Levenshtein distance [Lev66]. The edit distance computes the minimum number of edit operations (insertions, deletions, and substitutions) to transform from one string to another, and we denote the edit distance by *E*.

Table 6.2 shows an example of the similarity scores of the ORIGIN sequence field, the ORGANISM field (category of species) and the free-text DEFINITION of two scorpion venom records from the GenPept and Swiss-Prot database respectively.

Table 6.2: Similarity scores of Entrez records 1910194A and P45639

Field	1910194A	P45639	Score
ORIGIN	MCMPCFTTDHQMAR KCDDCCGGKGRGKC YGPQCLCR	MCMPCFTTDHQMARK CDDCCGGKGRGKCYG PQCLCR	1
ORGANISM	Leiurus quinquestriatus quinquestriatus	Leiurus quinquestriatus quinquestriatus	1
DEFINITION	chlorotoxin.	Chlorotoxin	0.92

6.3.3 Conjunctive Duplicate Rules

The overall similarity of two database records is determined from the similarities of individual fields of the records. Taking into consideration the correlations of fields and their similarity measures, a conjunctive clause of the matching criteria represents a duplicate relation. We call them duplicate rules, also known as merging rules. An example of a duplicate rule is:

Identical protein sequences ^ same length ^ same species → duplicate

The conjunctive clause is translated into a set of matching criteria and corresponding thresholds. This can be calculated by applying data type specific similarity functions (S for sequence similarity, N for numerical ratio and M for Boolean matching) on the sequence, sequence length and species fields respectively.

S(Seq)=1.0 ^ N(Seq Length)=1.0 ^ M(species)=1 → duplicate

If we encode the matching values as items, the rule takes the form of an association rule and we can easily apply association rule mining to induce models of the duplicate relations from dataset of known duplicates.

SE1.0 ^ LE1.0 ^ SP1 → duplicate

6.3.4 Association Mining of Duplicate Rules

The training dataset contains the similarity scores of pairs of records across the 9 criteria. To generate the items from the scores, we encode the values with field labels. For continuous values such as the sequence similarity scores which range from 0 to 1.0, the values are partitioned into equiwidth bins of 0.1. Hence, sequence similarity score item “SQ0.95” becomes “SQ0.9”. Figure 6.4 shows an input data set for association-based supervised classifier.

```
AAG39642 AAG39643 AC0.9 LE1.0 DE1.0 DB1 SP1 RF1.0 PD0 FT1.0 SQ1.0
AAG39642 Q9GNG8   AC0.1 LE1.0 DE0.4 DB0 SP1 RF1.0 PD0 FT0.1 SQ1.0
P00599   PSNJ1W   AC0.2 LE1.0 DE0.4 DB0 SP1 RF1.0 PD0 FT1.0 SQ1.0
P01486   NTSREB   AC0.0 LE1.0 DE0.3 DB0 SP1  RF1.0 PD0 FT1.0 SQ1.0
O57385   CAA11159 AC0.1 LE1.0 DE0.5 DB0 SP1 RF0.0 PD0 FT0.1 SQ1.0
S32792   P24663   AC0.0 LE1.0 DE0.4 DB0 SP1 RF0.5 PD0 FT1.0 SQ1.0
P45629   S53330   AC0.0 LE1.0 DE0.2 DB0 SP1 RF1.0 PD0 FT1.0 SQ1.0
```

Figure 6.4: Field labels from each pair of duplicates in training dataset

6.4 Performance Evaluation

The dataset is a combination of two set of records. The first data set consists of 520 scorpion toxin proteins retrieved from Entrez using the keywords “scorpion AND (venom OR toxin)”. The second set contains 780 snake PLA^2 venom proteins retrieved from Entrez using the keywords “serpentes AND venom AND PLA2”. The 1300 records were annotated separately by two biologists; 1328 duplicate pairs were identified collectively (Table 6.3).

Table 6.3: Different types of duplicate pair in training data set

Types of duplicates	Scorpion toxin	Snake PLA^2 toxin	Combined
1. Structural isoform	19	187	206
2. Duplicate	251	444	695
3. Cross-species duplicate	13	27	40
4. Sequence fragment	97	181	278
5. Cross-annotation variant	90	19	109
Total	470	858	1328

Experiments were performed on a Pentium-M 1.6GHz computer with 1GB of main memory, and running Windows XP. Association mining is carried out using CBA [LHM99] (Classification Based on Association) while other classifiers methods are evaluated using WEKA [HDW94].Figure 6.5 shows the accuracy results using different classifiers.

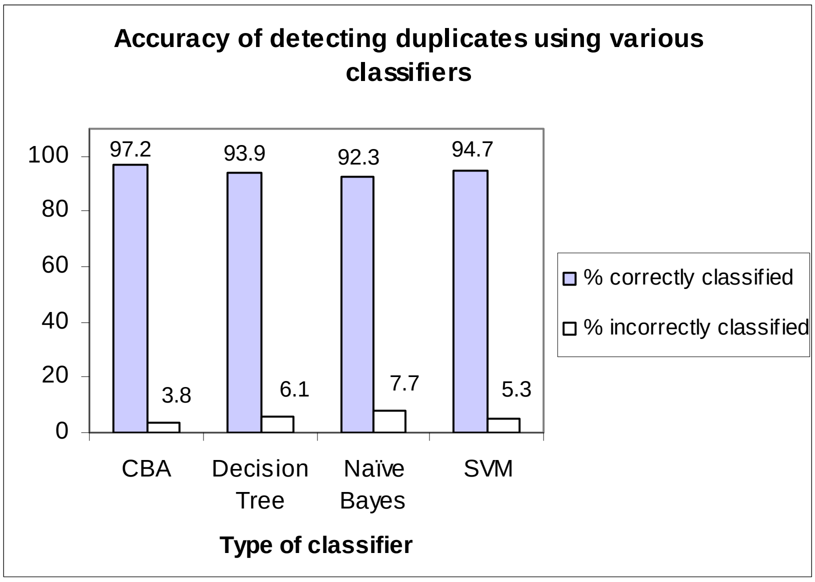


Figure 6.5: Accuracy of detecting duplicates using different classifiers

In general, association-based classifier yields better accuracy, achieving a positive predictive rate of 97.2%. Figure 6.6 shows the breakdown of the results into various types of duplicate relation. Cross-annotation variants are less predictive because variant sequences are usually considered as "similar" and the two biologists have different perspectives over what is considered "significantly similar". This fuzzy characteristic is inherent in biological data.

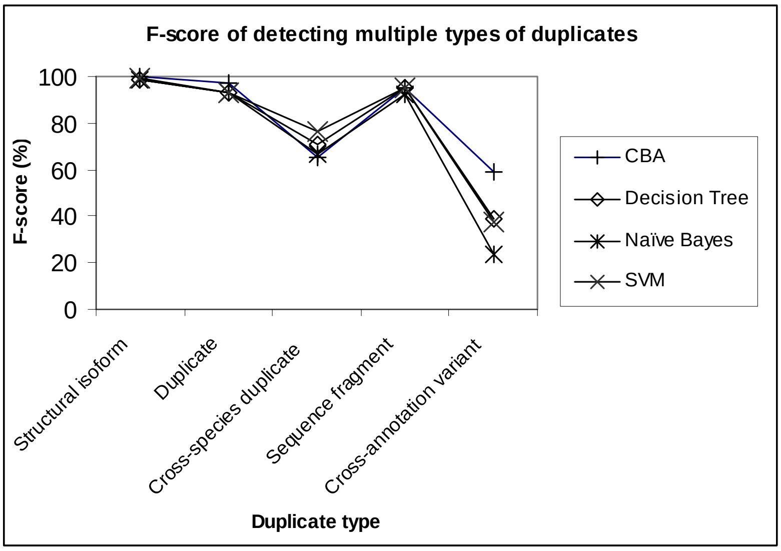


Figure 6.6: F-score of detecting different types of duplicates

Apart from the relative higher accuracy, the association mining approach has a practical advantage over other approaches; it does not require the negative data set of non-duplicates. First, it is difficult to collect a complete set of non-duplicates, and in addition, the input data would be highly skewed because the negative training set is naturally much larger than the positive counterparts. Table 6.4 shows part of the 181 duplicate rules induced from CBA.

Table 6.4: Examples of duplicate rules induced from CBA

Rule	Conf %	Sup %
1. M(PDB)=1 ^ M(Ref)=1.0 → Structural Isoform	100	9.5
2. M(Feature)=1 ^ M(PDB)=1 ^ E(Accession)=0.8 → Structural Isoform	100	7.8
3. M(Ref)=1 ^ M(Species)=1 ^ M(DB)=0 ^ E(Definition)=0.3 ^ S(Sequenc)=1 →Duplicate	100	5.0
4. M(PDB)=0 ^ E(Definition)=0.9 ^ M(Seq length)=1 ^ E(Accession)=0.8 → Variant	100	0.15
5. M(Species)=1 ^ M(Similarity)=0.9 → Fragment	100	3.0

For example, rule (1) indicates that a pair of sequence records from the same data source of PDB (meaning they are both translated from structural proteins) and contains the same references are structural isoforms. Rule (2) indicates that structural isoforms have identical features and their accession differs only slightly. It makes sense as structural duplicates representing different chains differ by a chain suffix, such as 1DJT_A and 1DJT_B. Rule (3) shows that identical sequences from the same species and relates to the same research articles, but from different data sources, are likely duplicates.

6.5 Concluding Section

In this chapter, we presented an application of correlation-based learning technique for duplicate detection. The chapter achieved preliminary contributions to duplicate detection for biological data. It explores scoring functions and criteria for matching sequence records. Also, it introduces a new method for modeling different types of duplicate relations using association rules and we compare with other classifiers including decision tree C4.5, Naïve Bayes and SVM. The duplicate rules identified from this work can be used for eliminating duplicates in protein sequence databases.

Chapter 7: Discussion

Every great mistake has a halfway moment, a split second when it can be recalled and perhaps remedied.

Pearl Buck
Author (1892 - 1973)

The information overload era result in a manifestation of low quality data in real-world databases. The demand for high quality data surges and opens new challenges for data cleaning. This thesis aims at tackling the data quality problem through an in-depth study of the data quality problem and the development of data cleaning techniques. We holistically addressed the problem of data artifacts in real-world biological databases and proposes 3 new general correlation-based data cleaning methods.

The completion of this research project made 4 specific contributions to the research in data cleaning as well as bioinformatics. Specific results and findings from each problem researched in this thesis are summarized in this chapter.

7.1 Review of Main Results and Findings

7.1.1 Classifications of Biological Data Artifacts

Chapter 3 of this thesis examines the varying types of artifacts in biological data. We observed that the data quality problem is a collective result of 11 types and 28 sub-types of artifacts at the field, record, single and multiple-database levels. It is also a combinatory problem of the bioinformatics that deals with the syntax and semantics of data collection, annotation, and storage, as well as the complexity of biological data. We developed both physical and conceptual classifications of these data artifacts; these classifications can be used as a “roadmap” for cleaning biological data. Representative examples of each type of artifacts are extracted from real-world biological databases and documented into an online catalogue called BioDArt (http://antigen.i2r.a-star.edu.sg/BioDArt/). To the best of our knowledge, this is the first complete study of biological data artifacts, with the objective of gaining holistic insights into the data quality problem and the adequacy of current data cleaning techniques.

Some artifacts can be addressed using existing data cleaning technique, while other more complicated artifacts require new methods. For example, these exists no known data cleaning method to resolve annotation errors, which affect 5% to 40% of the public protein and nucleotide sequences. Also, the problem of sequence redundancy is unlike the classical definition of duplicates in data cleaning; varying types of duplicate relations exist. The rest of the thesis is motivated at resolving these two data artifacts through new correlation-based data cleaning approaches.

7.1.2 Attribute Outlier Detection using ODDS

Chapter 4 focuses on the ODDS correlation-based detection method for attribute outliers. Unlike traditional class outlier detection research which consider outlier-ness as a global property applicable to all dimensions of the data set, our notion of attribute outlier-ness is a bivariate property of an attribute value and the subspace where it exhibits abnormal correlation. This is because for attribute outliers, rarity does not equate attribute outlier-ness but rather, the deviating correlation behaviour. Therefore, the ODDS algorithm involves finding both the attributes as well as the associated subspaces.

We also devised 3 new metrics *O-measure*, *Q-measure* and *O_f-measure* to quantify attribute outlier-ness. Experiments with synthetic data shown that O-measure is the most accurate while Q-measure is computationally less intensive. O_f-measure is devised for sparse data sets containing vast occurrences of rare attribute values which are not outliers. The number of attribute outliers differ from one dataset to another, depending on the noise level. Therefore, we developed an adaptive *Rate-of-change* factor to select optimal thresholds for distinguishing the outliers from non-outliers in any given data set. These automatic and data-dictated thresholds remove dependency on user-defined parameter. Because of the high time-cost of enumerating subspaces, we also introduced two strategies to filter subspaces that do not contain attribute outliers.

ODDS achieves an F-score of up to 88% in a synthetic data set for database tuples containing between 1 to 3 attribute outliers. Experiments with the UniProtKB/TrEMBL protein data set shown that ODDS achieve a positive predictive value (PPV) of up to 55% in detecting erroneous annotations.

7.1.3 Attribute Outlier Detection in XML using XODDS

Increasing biological databases are converted into XML formats in order to facilitate data exchange, including the protein databases. However, current outlier detection methods for relational data models are not directly adaptable to XML documents. Chapter 5 proposes XODDS - a four steps framework towards identifying attribute outliers in XML documents.

Besides utilizing correlations between attributes to adaptively identify attribute outlier, XODDS leverages on the hierarchical structure of the XML document to provide contextual information lacking in relational data, with the aim of improving both the effectiveness as well as efficiency of identifying attribute outliers. Specifically, 2 novel concepts of *correlated subspaces* and *aggregate attributes* in XML were introduced. Respectively, they reduce the time complexity of the attribute outlier method by separating the XML document into several natural partitions and enable summarization of group of nodes for data cleaning at higher level of abstractions.

We also develop for XODDS, the *x*O-measure and *x*Q-measure outlier scoring metrics which were adapted from O-measure and Q-measure. Experimental evaluation of *x*O-measure and *x*Q-measure with other correlation-based measures show that they significantly outperform other measures namely the Piatetsky-Shapiro rule interest, Interest factor, Jaccard coefficient, Hmeasure and Probability. XODDS also consistently performs better compared to the relational approach with F-scores of between 63%-86%. The introduction of aggregate attributes additionally identifies inherent attribute outliers.

When applied to the detectoin of annotation errors in UniProt/TrEMBL, XODDS attains a 97% positive predictive value (PPV), with significant improvement over ODDS.

7.1.4 Detection of Multiple Duplicate Relations

Chapter 6 of this thesis proposes a new approach to detect the varying types of duplicate relations. Unlike traditional duplication detection approaches, we consider the multi-facets of redundancy and develop a association rule induction method to model the various types of duplication. The method is used to identify 5 types of duplicates in biological data – duplicates, structural isoforms, cross-species duplicates, sequence fragments, and cross-annotation variants.

Experimental evaluation on a scorpion and snake venom protein data set with known duplicates shows that duplicate rules learned from association-based classifiers are capable of identifying up to 97.3% of the varying types of duplicates. Slight improvement is achieved over other classifiers, but the approach has practical advantage of requiring only the positive training set of duplicates.

7.2 Future Works

Overall, there remains several aspects of data cleaning that require further research. This section proposes two key research directions.

7.2.1 Biological Data Cleaning

Data quality is a multifactorial problem. In Chapter 3, we determined that for biological data, data quality problem is the combine effect of 11 types and 28 sub-types of data artifacts. To detect and to correct each type of artifact is an extensive data cleaning project of its own. The data cleaning methods proposed in this thesis focus on the detection of annotation errors and duplicates that cover only 2 of the known artifacts in the classification. Developing the detection methods already constitutes to more than 3 years of research. There are several other interesting research problems in biological data cleaning which have not been completely resolved. For instance, the problem of term disambiguation have also drawn increasing attention in the recent years. Moreover, only partial solutions have been developed. As with many other data cleaning problems for biological data, the difficulty of untangling the “web” of synonymy and homonymy in molecular entities stems from the inherent complexity of biology as an empirical science.

In addition, current approaches to data cleaning (including the methods proposed in this thesis) largely focus on the detection of the artifacts and not their correction, which in turn, requires new algorithms and methods. Clearly, the development of data cleaning techniques is at its infancy and it is becoming more critical in the bioinformatics domain as data continue to accumulate at an exponential rate and artifacts are proliferating among the diversified data sources, handicapping large-scale analysis.

Further work is needed to tackle the depreciating data quality problem in bioinformatics; a spectrum of data cleaning approaches addressing the assorted types of data artifacts from varying origins or sources, and affecting different parts of the databases is required. The classifications of biological data artifacts that we proposed in Chapter 3 of this thesis serve as a "roadmap" for the continuation of future work in biological cleaning research.

7.2.2 Data Cleaning for Semi-structured Data

Current works in data cleaning have primarily focused on structured databases to discover duplicate records and outliers. Semi-structured data models such as XML is rapidly proliferating as a new standard for data representation and exchange on the World Wide Web. As the world head towards the paper-less society, digital libraries containing unstructured data are also becoming increasingly popular for extracting information and text-mining.

On the other hand, the intrinsic structural differences between relational and XML or unstructured data models limit the direct adaptation of conventional data cleaning methods. Morever, the hierarchical structure of XML data provides additional semantic context that can be exploited to enhance the data cleaning method. We have shown in Chapter 5 that the hierarchical structure of XML enables the identification of semantic correlated subspaces within a XML document and the definition of aggregate attributes to summarize complex objects so that they can be compared at higher level of abstraction. And we demonstrate that the use of these two concepts significantly improve the accuracy of the outlier detection process. These are probably two examples of utilizing the structural models of XML as a means to enhance the data mining process. Further research is required to fully exploit the structural differences of XML and relational data to derive contextual information lacking in the latter and use it to improve our data mining techniques.

Bibliography

[ACG02] R. Ananthakrishna, S. Chaudhuri, and V. Ganti. Eliminating Fuzzy Duplicates in Data Warehouses. *VLDB*, pages 586-597, 2002.

[ADNB06] O. Arieli, M. Denecker, B. Nuffelen, and M. Bruynooghe. Computational methods for database repair by signed formulae. *Annals of Mathematics and Artificial Intelligence*, 46(1-2): 4-37(34), 2006.

[AIRR99] M. S. Almeida, M. Ishikawa, J. Reinschmidt, and T. Roeber. *Getting Started with Data Warehouse and Business Intelligence*. IBM Redbooks, 1999.

[AIS93] R.Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. *ACM SIGMOD*, pages 207-216, 1993.

[AMS$^+$97] S. F. Altschul, T. L. Madden, A. A. Schaeffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, 25(17):3389-3402, 1997.

[AY05] Aggarwal, C.C., Yu, P.S. An Effective and Efficient Algorithm for High-dimensional Outlier Detection. *VLDB Journal*, 14(2):211-221, 2005.

[BB96] P. Bork and A. Bairoch. Go hunting in sequence databases but watch out for the traps. *Trends in Genetics*, 12(10):425-427, 1996.

[BBA$^+$03] B. Boeckmann, A. Bairoch, R. Apweiler, M. –C. Blatter, A. Estreicher, E. Gasteiger, M. J. Martin, K. Michoud, C. O Donovan, and I. Phan. The SWISS-PROT protein knowledgebase and its supplement TrEMBL in 2003. *Nucleic Acids Research*, 31(Database issue):365-370, 2003.

[BC00] D. Barbará and P. Chen. Using the fractal dimension to cluster datasets. *ACM SIGKDD*, pages 260-264, 2000.

[BC01] P. Bork and R. Copley. The draft sequences. Filling in the gaps. *Nature*, 409(6822):818-820, 2001.

[BC03] L. E. Bertossi and J. Chomicki. Query answering in inconsistent databases. *Logics for Emerging Applications of Databases*, J. Chomicki, van der Meyden and G. Saake eds, Springer, pages 43-83, 2003.

[BFFR05] P Bohannon, W Fan, M Flaster, and R Rastogi. A cost-based model and effective heuristic for repairing constraints by value modification. *ACM SIGMOD*, pages 143-154, 2005.

[Bin93] M. Binns. Contamination of DNA database sequence entries with Escherichia coli insertion sequences. *Nucleic Acids Research*, 21:779-779, 1993.

[BK02] C. Borgelt and R. Kruse. Induction of association rules: Apriori implementation. *14th Conference on Computational Statistics*, pages 395-400, 2002.

[BK98] P. Bork and E. V. Koonin. Predicting functions from protein sequences–where are the bottlenecks? *Nature Genetics*, 18:313-318, 1998.

[BK04] V. Brusic and J. L. Y. Koh. Genetic Databases. *Mammalian Genomics*, A. Ruvinsky and J. A. Marshall-Graves ed., CABI Publishing, Wallingford, Chapter 16, 411-427, 2004.

[BKL$^+$06] D. A. Benson, I. Karsch-Mizrachi, D. J. Lipman, J. Ostell, and D. L. Wheeler. GenBank. *Nucleic Acids Research*, 34(Database issue):16-20, 2006.

[BKNS00] M. M. Breunig, H. P. Kriegel, R. T. Ng, and J. Sander. Lof: Identifying Density-based Local Outliers. *ACM SIGMOD*, 93-104, 2000.

[BL94] V. Barnett and T. Lewis. *Outliers in Statistical Data.* John Wiley and Sons, New York, 1994.

[BM03] M. Bilenko and R. J. Mooney. Adaptive duplicate detection using learnable string similarity measures. *ACM SIGKDD*, 39-48, 2003.

[BMBA00] A. G. Büchner, M. Baumgartenand M. D. Mulvenna, R. Böhm, and S. S. Anand. Data mining and XML: Current and future issues. *1st International Conference on Web Information Systems Engineering (WISE)*, pages 127-131, 2000.

[BMS97] S. Brin, R. Motwani, and C. Silverstein. Beyond market baskets: Generalizing association rules to correlations. *ACM SIGMOD Record*, 26(2):265-276, 1997.

[Bor01] P. Bork. Powers and Pitfalls in Sequence Analysis: The 70% Hurdle. *Genome Research*, 10(4):398-400, 2001.

[Bre99] S. E. Brenner. Errors in genome annotation. *Trends in Genomics (TIG)*, 15:132-133, 1999.

[BZSH99] V. Brusic, J. Zeleznikow, T. Sturniolo, E. Bono, and J. Hammer. Data cleansing for computer models: a case study from immunology, *6th International Conference on Neural Information Processing (ICONIP)*, 2:603-609, 1999.

[CBLJ04] D. P. A. Corney, B. F. Buxton, W. B. Langdon, and D. T. Jones. BioRAT: Extracting Biological Information from Full-length Papers, *Bioinformatics*, 20(17):3206-3213, 2004.

[CCD$^+$99] S. Ceri, S. Comai, E. Damiani, P. Fraternali, S. Paraboschi, and L. Tanca, XMLGL: A Graphical Language for Querying and Restructuring XML, *WWW.*, pages 93-109, 1999.

[CD04] J. S. Coker and E. Davies. Identifying adaptor contamination when mining DNA sequence data. *BioTechniques*, 31(2):194-198, 2004.

[CFR$^+$01] D. Chamberlin, D. Florescu, J. Robie, J. Simeon, and M. Stefanascu. XQuery: A query language for XML. *World Wide Web Consortium*, 2001. Available from http://wwww.w3.org/TR/xquery/.

[CGGM03] S. Chaudhuri, K. Ganjam, V. Ganti, and R. Motwani. Robust and efficient fuzzy match for online data cleaning, *ACM SIGMOD*, 313-324, 2003.

[CHDS05] A. M. Cohen, W. R. Hersh, C. Dubay and K. Spackman. Using co-occurrence network structure to extract synonymous gene and protein names from MEDLINE abstracts. *BMC Bioinformatics*, 6(1):103, 2005.

[Coh00] W. W. Cohen. Data Integration using similarity joins and a word-based information representation language. *Information Systems*, 26(8):607-633, 2001.

[CPW$^+$01] M. Cornell, N. W. Paton, S. Wu, C. A. Goble, C. J. Miller, P. Kirby, K. Eilbeck, A. Brass, A. Hayes, and S. G. Oliver. GIMS - A Data Warehouse for Storage and

Analysis of Genome Sequence and Functional Data. *2nd IEEE International Symposium on Bioinformatics and Bioengineering*, pages 15-22, 2001.

[Cri70] F. Crick. Central Dogma of Molecular Biology. *Nature*, 227:561-563, 1970.

[Cri58] F. H. C. Crick. On Protein Synthesis. *Annual symposium of the Society for Experimental Biology and Medicine*, XII:139-163, 1958.

[DA95] M. Dean and R. Allikmets. Contamination of cDNA libraries and expressed-sequence-tags databases. *American Journal of Human Genetics*, 57(5):1254-1255, 1995.

[DAB+05] N. Deshpande, K. J. Addess, W. F. Bluhm, J. C. Merino-Ott, W. Townsend-Merino, Q. Zhang, C. Knezevich, L. Xie, L. Chen, Z. Feng, R. K. Green, J. L. Flippen-Anderson, J. Westbrook, H. M. Berman, and P. E. Bourne. The RCSB Protein Data Bank: a redesigned query system and relational database based on the mmCIF schema. *Nucleic Acids Research*, 33(Database issue):233-237, 2005.

[DBBG00] C. Discala, X. Benigni1, E. Barillot, and G. Vaysseix. DBcat: a catalog of 500 biological databases. *Nucleic Acids Research*, 28(Database issue):8-9, 2000.

[DMM+03] P. Durand, C. Medigue, A. Morgat, Y. Vandenbrouck, A. Viari, and F. Rechenmann. Integration of data and methods for genome analysis. *Current Opinion in Drug Discovery and Development,* 6, 346-352, 2003

[Eck02] W. W. Eckerson. Achieving business success through a commitment to high quality data, *The Data Warehousing Institute Report Series*, No.101, Chatsworth, USA.

[EKV07] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios. Duplicate Record Detection: A Survey. *IEEE TKDE*, 19(1):1-16, 2007.

[Esk02] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S. A. Stolfo. Geometric Framework for Unsupervised Anomaly Detection: Detecting Intrusions in Unlabeled Data. *Applications of Data Mining in Computer Security,* D. Barbara, S. Jajodia (eds), Kluwer Academics Publishers, 77-102, 2002.

[Eve77] B. S. Everitt. *The analysis of Contigency Tables*, Chapman and Hall, 1977.

[EVE02] M. G. Elfeky, V. S. Verykios, A. K. Elmagarmid. TAILOR: A Record Linkage Tool Box. *IEEE ICDE*, pages 17-28, 2002.

[FCM+04] Z. Feng, L. Chen, H. Maddula, O. Akcan, R. Oughtred, H. M. Berman, and J. Westbrook. Ligand Depot: a data warehouse for ligands bound to macromolecules. *Bioinformatics*, 20(13):2153-2155, 2004.

[FHB+02] K. Fellenberg, N. C. Hauser, B. Brors, J. D. Hoheisel, and M. Vingron. Microarray data warehouse allowing for inclusion of experiment annotations in statistical analysis. *Bioinformatics*, 18(3):423-433, 2002.

[FPS99] U. M. Fayyad, . Pietetsky-Shapiro, and P. Smyth. *From data mining to knowledge discovery*. Advances in Knowledge Discovery and Data Mining, Fayyad, Piatetsky-Shapiro, Smyth and Uthurusamy *eds*, AAAI/MIT press, pages 1-34, 1996.

[HDW94] G. Holmes, A. Donkin, I. H. Witten. WEKA: a machine learning workbench. *Second Australian and New Zealand Conference on Intelligent Information Systems*, 357-361, 1994.

[GAA+00] R. Guigo, P. Agarwal, J. F. Abril, M. Burset, and J. W. Fickett. An assessment of gene prediction accuracy in large DNA sequences. *Genome Research*, 10:1631-1642, 2000.

[GAD+02] W.R. Gilks, B. Audit, D. De-Angelis, S. Tsoka, and C. A. Ouzounis. Modeling the percolation of annotation errors in a database of protein sequences. *Bioinformatics*, 18(12):1641-1649, 2002.

[Gal06] Y. Galperin. The Molecular Biology Database Collection: 2006 update. *Nucleic Acids Research*, 34(Database edition):3-5, 2006.

[GCB+97] J. Gray, S. Chaudhuri1, A. Bosworth1, A. Layman1, D. Reichart, M. Venkatrao, F. Pellow, and H. Pirahesh. Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals. *Data Mining and Knowledge Discovery*, 1(1):29-53, 1997.

[GFS+01] H. Galhardas, D. Florescu, D. Shasha, E. Simon, and C. A. Saita. Declarative Data Cleaning: Language, model, and algorithms. *VLDB*, pages 371-380, 2001.

[GFSS00] H. Galhardas, D. Florescu, D. Shasha, and E. Simon. AJAX: an extensible data cleaning tool. *ACM SIGMOD*, pages 590-602, 2000.

[GH97] R. D. Gardner, D. A. Harle. Fault resolution and alarm correlation in high-speed networks using database mining techniques. *International Conference on Information, Communications and Signal Processing (ICICS)*, 3:1423-1427, 1997.

[GHQ95] A. Gupta, V. Harinarayan, and D. Quass. Aggregate-Query Processing in Data Warehousing Environments. *VLDB*, pages 358-369, 1995.

[Gib99] G. Gibson. What works. Data warehouse: decision support solution reduces patient admissions, saves payer millions. *Health Management Technology*, 20:42-46, 1999.

[Gus97] D. Gusfield. *Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology*. Cambridge University Press, 1997.

[GW99] B. Ganter and R. Wille. Formal Concept Analysis. *Matheematical Foundations*, Springer, 1999.

[HCH04] B. He, K. C. Chang, J. Han. Discovering complex matchings across Web query interfaces: A correlation-mining approach. *ACM SIGKDD*, pages 148-157, 2004.

[HDR01] V. Hatzivassiloglou, P. A. Duboue, and A. Rzhetsky. Disambiguating proteins, genes, and RNA in text: a machine learning approach. *Bioinformatics*, 17(suppl 1):97-106, 2001.

[Heu99] K. I. Heuer. The development cycle of a pharmaceutical discovery chemoinformatics system. *Medical Research Review*, 19:209–221, 1999.

[Her95] M. Hernandez. A generation of band joins and the merge/purge problem. *Techical report CUCS-005-1995, Department of Computer Science, Columbia University*, 1995.

[HGP+04] K. G. Herbert, N. H. Gehani, W. H. Piel, J. T. L. Wang, and C. H. Wu. BIO-AJAX: An extensible framework for biological data cleaning. *Sigmod Record*, 33(2):51-57, 2004.

[HH99] R. J. Hilderman and H. J. Hamilton. *Knowledge discovery and interestingness measures: a survey*. Regina: Dept. of Computer Science, University of Regina, 1999.

[HKK03] J. Hosaka, J. L. Y. Koh, and A. Konagaya. Effect of utilizing terminology on extraction of protein-protein interaction information from biomedical literature. *10th Conference of the European Chapter of the Association for Computer Linguistics (EACL)*, pages 107–110, 2003.

[HLS99] K. Hu, Y. Lu, C. Shi. Incremental Discovering Association Rules: A Concept Lattice Approach. *PaKDD*, pages 109-113, 1999.

[HMY+02] X. Hou, M. Mrug, B. K. Yoder, E. J. Lefkowitz, G. Kremmidiotis, P. D'Eustachio, D. R. Beier, and L. M. Guay-Woodford. Cystin, a novel cilia-associated protein, is disrupted in the cpk mouse model of polycystic kidney disease. *Journal of Clinical Investment*, 109(4):533-540, 2002.

[HRM00] D. Hristovski, M. Rogac, and M. Markota. Data warehousing and OLAP in public health care. *American Medical Informatics Association Symposium 2000*, pages 369-373, 2000.

[HS95] A. M. Hernández and J. S. Stolfo. The Merge/Purge Problem for Large Databases. *ACM SIGMOD*, pages 127-138, 1995.

[HS98] A. H. Mauricio and J. S. Stolfo. Real-world Data is dirty: Data Cleansing and the Merge/Purge Problem. *Data Mining and Knowledge Discovery*, 2(1):9-27, 1998.

[HXD03] Z. He, X. Xu, and S. Deng. Discovering cluster-based local outliers. *Pattern Recognition Letters* 24(9-10):1641-1650, 2003.

[Inm93] W. H. Inmon. *Building the Data Warehouse*, Wiley-QED, New York, 1993.

[ITA+03] I. Iliopoulos, S. Tsoka, M. A. Andrade, A. J. Enright, M. Carroll, P. Poullet, V. Promponas, T. Liakopoulos, G. Palaios, C. Pasquier, S. Hamodrakas, J.

Tamames, A. T. Yagnik, A. Tramontano, D. Devos, C. Blaschke, A. Valencia, D. Brett, D. Martin D, C. Leroy, L. Rigoutsos, C. Sander, and C. A. Ouzounis. Evaluation of annotation strategies using an entire genome sequence. *Bioinformatics*, 19(6):717-726, 2003.

[JB99] J. M. Juran and G. A. Blanton. *Juran's Quality Handbook*. McGraw-Hill, 1999.

[JD98] A.K. Jain and R.C. Dubes. *Algorithms for clustering data*. Prentice Hall, Englewood Cliffs, NJ, 1988.

[JTH01] W. Jin, A. K. H. Tung, and J. Han. Mining Top-n Local Outliers in Large Databases. *ACM SIGKDD*, pages 293-298, 2001.

[JTS01] M. F. Jiang, S. S. Tseng, and C. M. Su. Two-phase clustering process for outliers detection. *Pattern Recognition Letters*, 22(6-7): 691-700, 2001.

[KAA$^+$05] C. Kanz, P. Aldebert, N. Althorpe, W. Baker, A. Baldwin, K. Bates, P. Browne, A. van den Broek, M. Castro, G. Cochrane, K. Duggan, R. Eberhardt, N. Faruque, J. Gamble, F. G. Diez, N. Harte, T. Kulikova, Q. Lin, V. Lombard, R. Lopez, R. Mancuso, M. McHale, F. Nardone, V. Silventoinen, S. Sobhany, P. Stoehr, M. A. Tuli, K. Tzouvara, R. Vaughan, D. Wu, W. Zhu and R. Apweiler. The EMBL Nucleotide Sequence Database. *Nucleic Acids Research*, 33(Database Issue):29-33, 2005.

[KB05] J. L. Y. Koh and V. Brusic. Bioinformatics Database Warehousing. *Bioinformatics Technologies*, Y. P. P. Chen ed., Springer, Chapter 3:45-62, 2005.

[KCD99] L. Kohn, J. Corrigan, and M. Donaldson. *To Err Is Human: Building a Safer Health System*, National Academy Press, 1999.

[KCH$^+$03] W. Kim, B. J. Choi, E. K. Hong, S. K. Kim, and D. Lee. A Taxonomy of Dirty Data. *Data Mining and Knowledge Discovery*, 7(1):81-99, 2003.

[KCN06] Y. Ke, J. Cheng, and W. Ng. Mining quantitative correlated patterns using an information-theoretic approach. *ACM SIGKDD*, pages 227 – 236, 2006.

[KHL+06] A. M. Khan, A. T. Heiny, K. X. Lee, K. N. Srinivasan, T. W. Tan, J. T. August, and V. Brusic. Large-scale analysis of antigenic diversity of T-cell epitopes in dengue virus. *BMC Bioinformatics*, 7(Suppl 5):S4, 2006.

[KHRB96] P. G. Korning, S. M. Hebsgaard, P. Rouze, and S. Brunak. Cleaning the GenBank, Arabidopsis thailana data set. *Nucleic Acids Research*, 24, 316–320, 1996.

[Kim96] R. Kimball. Dealing with Dirty Data. *DBMS Online*, www.dbmsmag.com/9609d14.htm, 1996.

[KKSL+04] A. Kasprzyk, D. Keefe, D. Smedley, D. London, W. Spooner, C. Melsopp, M. Hammond, P. Rocca-Serra, T. Cox, and E. Birney. EnsMart: A Generic System for Fast and Flexible Access to Biological Data. *Genome Research*, 14(1):160-169, 2004.

[KKS+04] J. L. Y. Koh, S. P. T. Krishnan, S. H. Seah, P. T. J. Tan, A. M. Khan, M. L. Lee, and V. Brusic. BioWare: A framework for bioinformatics data retrieval, annotation and publishing. *ACM SIGIR Workshop on Search and Discovery in Bioinformatics (SIGIRBIO)*, 2004.

[KL05] N. Kaplan and M. Linial. Automatic detection of false annotations via binary property clustering. *BMC Bioinformatics*, 6:46, 2005.

[KLB05] J. L. Y. Koh, M. L. Lee, and V. Brusic. A classification of biological data artifacts, in *ICDT Workshop on Database Issues in Biological Databases*, pages 53-57, 2005.

[KLHA07] J. L. Y. Koh, M. L. Lee, W. Hsu and W. T. Ang. Correlation-based Outlier Detection in XML, *SIGKDD*, 2007 [Submitted]

[KLHL07] J. L. Y. Koh, M. L. Lee, W. Hsu and K. T. Lam. Correlation-based Detection of Attribute Outliers, *DASFAA*, 2007 [Accepted]

[KLK+04] J. L. Y. Koh, M. L. Lee, A. M. Khan, P. T. J. Tan, and V. Brusic. Duplicate Detection in Biological Data using Association Rule Mining, *ECML/PKDD*

Workshop on Data Mining and Text Mining for Bioinformatics, pages 35-41, 2004.

[KM03] J. Kubica and A. Moore. Probabilistic Noise Identification and Data Cleaning. *ICDE*, pages 131-138, 2003.

[KN98] E. M. Knorr and R. T. Ng. Algorithms for Mining Distance-Based Outliers in Large Datasets, *Proc. of the 24th International Conference on Very Large Data Bases (VLDB)*, pages 392-403, 1998.

[KN99] E. M. Knorr and R. T. Ng. Finding Intensional Knowledge of Distance-based Outliers. *VLDB*, pages 211-222, 1999.

[KNT00] E. M. Knorr, R. T. Ng, V. Tucakov. Distance-based Outliers: Algorithms and Applications. *VLDB Journal*, 8:237-253, 2000.

[KO02] S. Kuznetsov and S. Obiedkov. Comparing Performance of Algorithms for Generating Concept Lattices. Journal of Experimental & Theoretical Artificial Intelligence, 14, 189-216, 2002.

[KSZ01] P. D. Karp, S. Paley, and J. Zhu. Database verification studies of Swiss-Prot and GenBank. *Bioinformatics*, 17(6):526-532, 2001.

[ŁCS^{+}04] M. ŁoŚ, A. CzyŻ, E. Sell, A. WÊgrzyn, P. Neubauer, and G. WÊgrzyn. Bacteriophage contamination: is there a simple method to reduce its deleterious effects in laboratory cultures and biotechnological factories? *J Appl Genetic*, 45(1):111-120, 2004.

[LDB^{+}04] Leinonen, R. Diez, F. G. Binns, D. Fleischmann, W. Lopez, and R. Apweiler. UniProt Archive. *Bioinformatics*, 20, 3236–3237.

[Lev66] V. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics – Doklady 10*, 10:707-710, 1966.

[LHK04] M. L. Lee, W. Hsu, and V. Kothari. Cleaning up the spurious links in data. *IEEE Intelligent Systems: Special issue on Data and Information Cleaning and Preprocessing.* 19(2):28-33, 2004.

[LHM99] B. Liu, W. Hsu, Y. Ma. Pruning and summarizing the discovered associations. *ACM SIGKDD*, 125-134, 1999.

[LKCH03] Y. K. Lee, W. Y. Kim, Y. D. Cai, and J. Han. Comine: Efficient mining of correlated patterns. *IEEE ICDM*, pages 581- 584, 2003.

[LKP92] R. Lopez, T. Kristensen and H. Prydz. Database contamination. *Nature*, 355(6357):211, 1992.

[LKSV92] E. D. Lamperti, J. M. Kittelberger, T. F. Smith, and L. Villa-Komaroff. Corruption of genomic databases with anomalous sequence. *Nucleic Acids Research*, 20(11):2741–2747, 1992.

[LLH04] R. Lu, M. L. Lee, W. Hsu. Using interval association rules to identify dubious values. *Advances in Web-Age Information Management, pages* 528-538, 2004.

[LLL00] M. L. Lee, T. W. Ling, and W. L. Low. IntelliClean: a knowledge-based intelligent data cleaner. *ACM SIGKDD*, pages 290-294, 2000.

[LLLK99] M. L. Lee, H. Lu, T. W. Ling, and Y. T. Ko. Cleansing Data for Mining and Warehousing, *DEXA*, 751-760, 1999.

[LNZA06] R. Leinonen, F. Nardone, W. Zhu, and R. Apweiler. UniSave: the UniProtKB Sequence/Annotation Version database. *Bioinformatics*, 22(10):1284-1285, 2006.

[LSM99] W. Lee, S. J. Stolfo, and K. Mok. Data Mining in Work Flow Environments: Experiences in Intrusion Detection. *ACM SIGKDD*, 1999.

[LTLL02] W. L. Low, W. H. Tok, M. L. Lee, and T. W. Ling. Data Cleaning and XML : The DBLP Experience. Poster in *IEEE ICDE*, 2002. (full paper in www-appn.comp.nus.edu.sg/~esubmit/search/techrep_03.cgi?id=techrep;TRA1/03)

[LV03] P. Lyman and H. R. Varian. *How Much Information*, http://www.sims.berkeley.edu/how-much-info-2003, 2003.

[May78] J. A. Mayo. A comparison of methods for detecting bacteriophage contamination of tissue culture sera. *In Vitro*, 14:413-417, 1978.

[Met05] M. L. Metzker. Emerging technologies in DNA sequencing. *Genome Research*, 15:1767-1776, 2005.

[ME96] A. E. Monge and C. P. Elkan. The field matching problem: Algorithms and applications. *SIGMOD workshop on research issues on knowledge discovery and data mining*, pages 267-270, 1996.

[ME97] A. Monge and C. Elkan. An efficient domain-independent algorithm for detecting approximately duplicate database records. *Data mining and knowledge discovery*, 1997.

[MGB99] C. Miller, J. Gurd, and A. Brass. A RAPID algorithm for sequence database comparisons: application to the identification of vector contamination in the EMBL databases. *Bioinformatics*, 15(2):111-121, 1999.

[MNF03] H. Müller, F. Naumann, and J. Freytag. Data Quality in Genome Databases. *International Conference on Information Quality*, pages 269-284, 2003.

[MML01] A. Marcus, J. I. Maletic, K. Lin. Ordinal association rules for error identification in data sets. *ACM CIKM*, pages 589 - 591, 2001.

[MT01] V. M. Markowitz and T. Topaloglou. Applying data warehouse concepts to gene expression data management. *Bioinformatics and Bioengineering Conference (BIBE)*, 65-72, 2001.

[NW03] D. W. Nebert and H. M. Wain. Update on human genome completion and annotations: Genome nomenclature. *Human Genomics*, 1(1): 66-71, 2003

[NW70] S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequences of two proteins. *Journal of Molecular Biology*, 48:443-453, 1970.

[OH98] C. G. Overton and J. Haas. Case-Based Reasoning Driven Gene Annotation. *Computational Methods in Molecular Biology*. Elsevier Science, 32:65-86, 1998.

[Orr98] K. Orr. Data Quality and Systems Theory. *Communications of the ACM*, 41(2): 66-71, 1998.

[OS90] K. Osatomi and H. Sumiyoshi. Complete nucleotide sequence of dengue type 3 virus genome RNA. *Virology*, 176:643-647, 1990.

[OSGT06] K. Okubo, H. Sugawara, T. Gojobori, and Y. Tateno. DDBJ in preparation for overview of research activities behind data submissions. *Nucleic Acids Research*, 34(Database issue):6-9, 2006.

[PCG$^+$04] R. M. Podowski, J. G. Cleary, N. T. Goncharoff, G. Amoutzias, W. S. Hayes. AZuRE, a scalable system for automated term disambiguation of gene and protein names. Proceedings of *IEEE Computational Systems Bioinformatics Conference (CSB) 2004*, 415- 424, 2004.

[Pia91] G. Piatetsky-Shapiro. Discovery, analysis and presentation of strong rules. Knowledge Discovery in Databases, G. Piatetsky-Shapiro and W. Frawley eds, MIT Press, Cambridge, MA: 2299-2480, 1991.

[PHBR04] H. Pospisil, A. Herrmann, R. H. Bortfeldt, and J. G. Reich. EASED: Extended Alternatively Spliced EST Database. *Nucleic Acids Research*, 32(Database issue):70–74, 2004.

[PKGF03] S. Papadimitriou, H. Kitagawa, P. B. Gibbons, and C. Faloutsos. LOCI: Fast Outlier Detection using the Local Correlation Integral. *IEEE ICDE*, pages 315-326, 2003.

[PM01] K. D. Pruitt and D. R. Maglott. RefSeq and LocusLink: NCBI gene-centered resources. *Nucleic Acids Research*, 29(1):137-140, 2001.

[Poe96] V. Poe. *Building a Data Warehouse for Decision Support*. Prentice Hall PTR, 1996.

[PPKG03] T. Palpanas, D. Papadopoulos, V. Kalogeraki, and D. Gunopulos. Distributed deviation detection in sensor networks. *SIGMOD Record*, 4(32):77-82, 2003.

[PWL01] F. Pachet, G. Westermann, D. Laigre. Musical data mining for electronic music distribution. *Web Delivering of Music*, pages 101- 106, 2001.

[PWN06] S. Puhlmann, M. Weis, F. Naumann. XML Duplicate Detection Using Sorted Neighborhoods. *EDBT*, pages 773-791, 2006.

[RAMC97] B. L. Roberts, M. K. Anthony, E. A. Madigan, and Y. Chen. Data Management: Cleaning and Checking. *Nursing Research*, 46(6):350-352, 1997.

[RD00] E. Rahm and H. H. Do. Data Cleaning: Problems and Current Approaches *IEEE Technical Bulletin on Data Engineering*, 23(4): 3-13, 2000.

[Red04] T. Redman. Data: An Unfolding Quality Disaster. *DM Review*, August issue, 2004.

[RH01] V. Raman and J. M. Hellerstein. Potter's wheel: an interactive data cleaning system, *VLDB*, pages 381-390, 2001.

[RL87] P. J. Rousseeuw and A. M. Leroy. *Robust Regression and Outlier Detection*. John Wiley and Sons, 1987.

[RRK00] S. Ramaswamy, R. Rastogi, and S. Kyuseok. Efficient Algorithm for Mining Outliers from Large Data Sets. *ACM SIGMOD*, 427-438, 2000.

[RRP04] D. Ren, I. Rahal, W. Perrizo. A Vertical Outlier Detection Algorithm with Clusters as By-Product. *IEEE ICTAI*, 22-29, 2004.

[RRPS04] D. Ren, I. Rahal, W. Perrizo, K. Scott. A vertical distance-based outlier detection method with local pruning. *ACM CIKM*, 279-284, 2004.

[PTM05] K. D. Pruitt, T. Tatusova, and D. R. Maglott. NCBI Reference Sequence (RefSeq): a curated non-redundant sequence database of genomes, transcripts and proteins. *Nucleic Acids Research*, 33(Database issue):501-504, 2005.

[SB02] S. Sarawagi and A. Bhamidipaty. Interactive deduplication using active learning. *ACM SIGKDD*, pages 269-278, 2002.

[SBB+00] R. Stevens, P. Baker, S. Bechhofer, G. Ng, A. Jacoby, N. W. Paton, C. A. Goble, and A. Brass. TAMBIS: transparent access to multiple bioinformatics information sources. *Bioinformatics*, 16(2):184-185, 2000.

[SC05] L. Shi and F. Campagne. Building a protein name dictionary from full text: a machine learning term extraction approach. *BMC Bioinformatics*, 6(1):88, 2005.

[Sch98] R. Scheese. Data warehousing as a healthcare business solution. *Healthcare Financial Management*, 52(2):56-59, 1998.

[SCH$^+$98] L. Singh, B. Chen, R. Haight, P. Scheuermann, and K. Aoki. A robust system architecture for mining semi-structured data. *ACM SIGKDD*, 329-333, 1998.

[SEOK96] G. D. Schuler, J. A. Epstein, H. Ohkawa, J. A. Kans. Entrez: molecular biology database and retrieval system. *Methods Enzymol.* 266:141-162, 1996.

[SFM+99] G. A. Seluja, A. Farmer, M. McLeod, C. Harger and P. A. Schad. Establishing a method of vector contamination identification in database sequences. *Bioinformatics*, 15(2):106-110, 1999.

[SGT+02] K. N. Srinivasan, P. Gopalakrishnakone, P. T. Tan, K. C. Chew, B. Cheng, R. M. Kini, J. L. Y. Koh, S. H. Seah and V. Brusic. SCORPION, a molecular database of scorpion toxins. *Toxicon* , 40:23-31, 2002.

[She97] D. Shenk. *Data Smog: surviving the information glut.* New York, Harper and Collins, 1997.

[SHX+05] S. P. Shah, Y. Huang, T. Xu, M. M. S. Yuen, J. Ling, and B. F. F. Ouellette. Atlas - a data warehouse for integrative bioinformatics. *BMC Bioinformatics*, 6:34, 2005.

[SKB00] C. Schönbach, P. Kowalski-Saunders and V. Brusic. Data warehousing in molecular biology. *Briefings in Bioinformatics* 1, 190-198, 2000.

[SS03] R. Sorek and H. M. Safer. A novel algorithm for computa-tional identification of contaminated EST libraries. *Nucleic Acids Research*, 31(3):1067-1074, 2003.

[SSU96] A. Silberschatz, M. Stonebraker, and J. Ullman. Database research: Achievements and opportunities into the 21st century. *SIGMOD Record*, 25(1):52, 1996.

[Ste03] L. D. Stein. Integrating biological databases. *Nature Reviews Genetics*, 4(55):337-345, 2003.

[SW81] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147:195-197, 1981.

[Ten03] C. M. Teng. Applying noise handling techniques to genomic data: A case study. *IEEE ICDM*, pages 743- 746, 2003.

[Ten04] C. M. Teng. Polishing Blemishes: Issues in Data Correction. *IEEE Intelligent Systems*, 19, 2:34-39, 2004.

[Tha99] T. A. Thanaraj. A clean data set of EST-confirmed splice sites from Homo sapiens and standards for clean-up procedures. *Nucleic Acids Research*. 27(13):2627-2637, 1999.

[TKB03] P. T. J. Tan, A. M. Khan, and V. Brusic. Bioinformatics for venom and toxin sciences. *Briefings in Bioinformatics*, 4:53-62, 2003.

[TKS02] P. N. Tan, V. Kumar, and J. Srivastava. Selecting the right interestingness measure for association patterns. *ACM SIGKDD*, pages 32-41, 2002.

[TM99] T. A. Tatusova, and T. L. Madden. BLAST 2 Sequences, a new tool for comparing protein and nucleotide sequences. *FEMS Microbiology Letters*, 174:247–250, 1999.

[VCEK05] J. Van den Broeck, S. A. Cunningham, R. Eeckels, and K. Herbst. Data Cleaning: Detecting, Diagnosing, and Editing Data Abnormalities. *PLoS Med.* 2(10):e267, 2005.

[VVS$^+$00] P. Vassiliadis, Z. Vagena, S. Skiadopoulos, N. Karayannidis and T. Sellis. ARKTOS: A tool for data cleaning and transformation in data warehouse Environments. *IEEE Data Engineering Bulletin*, 23(4):42-47, 2000.

[WAB$^+$06] C. H. Wu, R. Apweiler, A. Bairoch, D. A. Natale, W. C. Barker, B. Boeckmann, S. Ferro, E. Gasteiger, H. Huang, R. Lopez, M. Magrane, M. J. Martin, R. Mazumder, C. O'Donovan, N. Redaschi, and B. Suzek. The Universal Protein Resource (UniProt): an expanding universe of protein information. *Nucleic Acids Research*, 34(Database issue):187-191, 2006.

[WDS$^+$93] O. White, T. Dunning, G. Sutton, M. Adams, J. C. Venter and C. Fields. A quality control algorithm for DNA sequencing projects. *Nucleic Acids Research*. 21(16):3829-3838, 1993

[Whe04] M. Wheatley. Operation Clean Data. *CIO Magazine*, Jul. 1, 2004.

[Wij05] J. Wijsen. On condensing database repairs obtained by tuple deletions. Proceedings. *DEXA*, pages 849- 853, 2005.

[Wil82] R. Wille. Reconstructing Lattice Theory: an Approach Based on Hierarchies of concepts. *Ordered sets*, Reidel, 1982.

[WKA04] D. Wieser, E. Kretschmann, and R. Apweiler. Filtering erroneous protein annotation. *Bioinformatics*, 20(Suppl. 1):342-347, 2004.

[WKM93] Wang, R., Kon, H. & Madnick, S., Data quality requirements analysis and modelling, *IEEE ICDE*, pages 670-677, 1993.

[WM89] Y. R. Wang and S. E. Madnick. The Inter-Database instance identification problem in integrating autonomous systems. *IEEE ICDE*, pages 46-55, 1989.

[WMN01] J. A. White, L. J. Maltais and D. W. Nerbert. An increasingly urgent need for standardised gene nomenclature. *Nature Genetics*, 2001.

[WN05] M. Weis and F. Naumann. DogmatiX tracks down duplicates in XML. *ACM SIGMOD*, pages 431-422, 2005.

[Wong01] L. Wong. Bioinformatics Integration Simplified: The Kleisli Way. *Frontiers in Human Genetics: Diseases and Technologies*, chapter 6:79-90. 2001

[WSF95] R. Wang, H. Kon and S. Madnick, A framework for analysis of data quality research, *IEEE TKDE*, 7(4):623-640, 1995.

[WYH$^+$03] C. H. Wu, L.-S. L. Yeh, H. Huang, L. Arminski, J. Castro-Alvear, Y. Chen, Z. Hu, P. Kourtesis, R. S. Ledley, B. E. Suzek. The Protein Information Resource. *Nucleic Acids Research*, 31(1):345-347, 2003.

[YA03] H. Yu and E. Agichtein. Extracting synonymous gene and protein terms from biological literature. *Bioinformatics*, 19(supp. 1): 340-349, 2003.

[YLL03] L. Yi, B. Liu, and X. Li. Eliminating Noisy Information in Web Pages for Data Mining. *ACM SIGKDD*, pages 296-305, 2003

[Zak04] M. J. Zaki: Mining Non-Redundant Association Rules. *Data Mining and Knowledge Discovery,* 9(3): 223-248, 2004.

[ZPOL97] M. J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li. New algorithms for fast discovery of association rules. *Proc. 3rd Intl. Conf. KDD*, page 283-296, 1997.

[ZLAE02] E. M. Zdobnov, R. Lopez, R. Apweiler, T. Etzold. The EBI SRS server—new features. *Bioinformatics*, 18:1149–1150, 2002.

[ZW04] X. Zhu and X. Wu. Class Noise vs. Attribute Noise: A Quantitative Study of their Impacts. *Artificial Intelligence Review*, 22(3):177-210, 2004.

[ZZS$^+$04] G. D. Zhou, J. Zhang, J. Su, D. Shen and C. L. Tan. Recognizing names in biomedical texts: a machine learning approach. *Bioinformatics*, 20(7):1178-1190, 2004.