BIODQ: A MODEL FOR DATA QUALITY ESTIMATION AND MANAGEMENT IN BIOLOGICAL DATABASES

By

ALEXANDRA MARTÍNEZ

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

2007

To my parents

ACKNOWLEDGMENTS

I thank God for guiding my life path. I thank my parents for their unconditional support and love throughout all my life, my husband for his love, patience, and help during my doctoral studies, and my daughter for being my inspiration. I specially thank the chair, co-chair, and members of my supervisory committee for their valuable mentoring. I also thank the participants of our research study; and our collaborators from the University of Florida Interdisciplinary Center for Biotechnology Research, the University of Florida Health Science Center Libraries, and the National Center for Biotechnology Information.

TABLE OF CONTENTS

page

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

API	Application Programming Interface.
AUC	Area under the ROC curve.
BLAST	Basic Local Alignment Search Tool.
CLOB	Character Large Object.
DBMS	Database Management System.
DDBJ	DNA Data Bank of Japan.
DMSI	Data Manager Service Interface.
DTD	Document Type Definition.
EMBL	European Molecular Biology Laboratory.
EUtils	Entrez Programming Utilities.
FIFO	First In, First Out.
GenBank	Genetic sequence database.
HQ	High Quality.
IQ	Information Quality.
IQR	Interquartile range.
JDBC	Java Database Connectivity.
LQ	Low Quality.
LRU	Least Recently Used.
MMSI	Metadata Manager Service Interface.
MRU	Most Recently Used.
NCBI	National Center for Biotechnology Information.
QEM	Quality Estimation Model.
QM	Quality Metadata.
QMA	Quality Metadata Architecture.

RefSeq	Reference Sequence collection.
ROC	Receiver operating characteristic.
SAX	Simple API for XML.
SQL	Structured Query Language.
W3C	World Wide Web Consortium.
XML	Extensible Markup Language.

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

BIODQ: A MODEL FOR DATA QUALITY ESTIMATION AND MANAGEMENT IN BIOLOGICAL DATABASES

By

Alexandra Martínez

December 2007

Chair: Joachim Hammer
Major: Computer Engineering

We present BIODQ, a model for estimating and managing the quality of biological data in genomics repositories. BIODQ uses our new Quality Estimation Model (QEM) which has been implemented as part of the Quality Management Architecture (QMA). The QEM consists of a set of quality dimensions and their quantitative measures. The QMA combines a series of software components that provide support for the integration of the QEM with existing biological repositories. We describe a research study conducted among biologists, which provides insights into the process of quality assessment in the biological context, and is the basis of our evaluation. The evaluation results show that the QEM dimensions and estimations are biologically-relevant and useful for discriminating high quality from low quality data. Additionally, the evaluation performed on a subset of the National Center for Biotechnology Information's databases validates the benefits of QMA as a quality-aware interface to genomics repositories. We expect BIODQ to benefit biologists and other users of genomics repositories by providing them with accurate information about the quality of the information that is returned as part of their queries.

CHAPTER 1
INTRODUCTION

The rapid accumulation of biological information as well as their widespread usage by scientists to carry out research is posing new challenges to determine and help manage the quality of data in public biological repositories. Genbank [35, 5], RefSeq [39, 49], and Swissprot [58, 7] are prominent examples of public repositories extensively used by genomics researchers and practitioners, bioinformaticians, and biologists in general. In the least, analysis and processing of low-quality data results in wasted time and resources. In the worst case, the usage of low-quality data may lead scientist to false conclusions or inferences, thus hampering scientific progress.

Several quality models and assessment methodologies have been proposed in the literature, but most are utilized in the context of enterprise data warehousing and aim to solve quality problems existing in the business domain. These methodologies do not fit naturally into the genomics context because biological data is more complex and less structured than typical business data. In addition, the increasing data generation and usage rates limit the kind of quality assessments that can realistically be performed. For example, a common approach for assessing information quality has been to gather user appraisals about the perceived quality of information products or processes, mostly via questionnaires. This approach, as any other manual-based approach of quality assessment, turns impractical in the context of genomics due to its lack of scalability and inability to produce timely quality estimates. We therefore believe that there is a need for automated quality assessment techniques that provide users of genomics data sources with objective and quantitative estimates of the quality of available data.

1.1 The Need for Quality Information in Biological Data Sources

1.1.1 Biological Data Sources Currently do not Manage Quality Well

We studied some of the popular public repositories of biological data with the purpose of discovering how they manage quality. Most of the problems we found were common to all the repositories; hence we focused our work on the databases of the National Center for Biotechnology Information (NCBI) [61] because of their widespread use by the scientific community. The three major problems found related to quality are described next.

1.1.1.1 Shortage of quality information

Currently, biological data sources provide minimal information about the quality of the stored data. Some repositories offer base-calling scores, which are quality indicators of the sequence data solely. Typically, however, genomic records contain a significant amount of annotations about the sequence data, which should be accounted for in a comprehensive evaluation of records.

Several challenges must be overcome when addressing the shortage of quality information provided by the data sources. First, comprehensive quality assessments need to be formulated, which consider the entire contents of a record (i.e., sequence and annotations). Second, different quality aspects of the stored data should be available in order to accommodate the large variation in usage and quality perception by users of the data sources. Consequently, quality must be evaluated from a multidimensional perspective. Third, a mechanism to represent and store quality information about the underlying biological data needs to be devised.

1.1.1.2 High data generation to curation ratio

Most pubic biological repositories have a curation process in place, which consists in cleaning, standardizing, and annotating the submitted data with the aim of improving its value and quality. Even though this curation process has been partially automated, a significant amount

of human effort (and time) is still required. On the other hand, large amounts of biological data coming from multiple sequencing centers and laboratories are loaded into the repositories on a daily basis. Hence, the ratio of data generation to data curation is increasing, and this trend will continue as sequencing technologies are improved. For this reason, most biological data sources publish their newly acquired data before it can be completely curated, thus raising concern over the quality of available data.

One approach for addressing the high data generation to curation ratio problem is to work towards a fully automated curation process of the submitted data. However, until this becomes a viable option, an alternative approach is to provide indicators of the quality of the available data that help users recognize data of different curation (and quality) level.

1.1.1.3 Lack of quality-driven interfaces

Current query interfaces of biological data sources do not support user-specified quality criteria as part of queries. Without such capability, the identification of high-quality records from the query results becomes a time-consuming task even for experienced users. While experienced users can generally glance at a record and roughly estimate its quality level, when a query retrieves a large number of records, examining each record individually is not convenient. Moreover, users who are new to one of these repositories would need to become familiar with the implicit quality indicators embedded in the data, before they can interpret and use them in a quality assessment. Not to mention that criteria used to evaluate the retrieved records are subjective and largely depend on user expertise. Hence, an automated way to present the query results ranked by quality score would be preferred.

We believe that biological data sources should provide query interfaces that allow users to (i) selectively request quality information over the retrieved records, (ii) filter out data whose

quality does not meet the expectations specified by the user, and (iii) order query results with respect to a given quality dimension.

1.1.2 Benefits of Adding Quality Information

The benefits of adding quality information to existing biological repositories are numerous and their impact is broad. Here we outline some of the major benefits we foresee.

First, the value and utility of existing repositories will be enhanced by providing users with quality information about the retrieved data, which will in turn help the users decide what data best fits their specific needs. Second, biologists and other users of current repositories will be able to work more effectively when using a quality-aware interface that allows them to filter out query results below a certain quality threshold, and to rank the retrieved data based on different quality scores. This will aid users to quickly discriminate high quality records from a candidate set, and decide what data best fits their specific needs. Third, the data curation process will be facilitated by providing preliminary estimates for the quality of records submitted to the database, which can in turn help curators prioritize records for further revision.

1.2 Motivating Example

The Reference Sequence (RefSeq) collection [39, 49], a public database of biological sequences maintained by the National Center for Biotechnology Information, is considered one of the high quality databases currently available. NCBI's RefSeq database consists of curated, non-redundant nucleotide and protein sequences from numerous organisms (3774 as of Release 19) [49]. The curation process is performed by collaborating groups and by NCBI staff and records are annotated with a curation status: model, predicted, inferred, provisional, validated

and reviewed. 'Reviewed' and 'Validated' are the two highest curation levels, followed by 'Provisional', and then by 'Predicted', 'Inferred', and 'Model'[1].

Some could argue that the shortage of quality information can be easily solved by using the curation status terms as quality categories of the data. While this could be used as a first approach to categorize records from the RefSeq database, it does not convey a general and comprehensive solution. First, this quality categorization derived from RefSeq would neither apply to databases that have a different curation process in place, nor to those whose curation process is rather unstructured, making it hard to even identify curation levels. Second, relying on a curation status as a quality indicator assumes the existence of a curation process, but such assumption may not hold true for some data sources (including GenBank) which are mainly archival datasets. Third, records at the same curation level may not necessarily be considered of similar quality by domain experts (according to our research study, described in Appendixes A and B). For example, the RefSeq records identified by accessions NM_181070 and NM_174874 are both annotated with the 'Provisional' curation status (an intermediate curation level); however, the former was classified as a 'good' quality record while the latter was classified as a 'poor' quality record, by one of the research participants. This shows that the curation level alone is not sufficient to determine the quality of data stored in a biological repository, and serves to illustrate that assessing the quality of data in the biological context is a difficult problem.

1.3 Our Contributions

The main contributions of this work are:

- The identification of a set of measurable quality dimensions fit for genomic data.

[1] Information obtained partially from NCBI collaborators and partially from documentation available at the RefSeq website http://www.ncbi.nlm.nih.gov/RefSeq/key.html#status.

- The formulation of quantitative measures for the quality dimensions, which can be systematically computed in a semistructured data model.
- The definition of a core set of quality-aware maintenance and query operations for a semistructured data model.
- The design, implementation, and validation of a quality management architecture, which enables the integration of our quality model with existing biological data sources.
- The development of a research study among domain experts (biologists), which is the basis for the experimental evaluation of our model.
- The evaluation of the significance and usefulness of the chosen quality dimensions and measures in assessing the quality of biological data.
- The evaluation of the usefulness of the quality management architecture with respect to its functional capabilities and operational costs.

CHAPTER 2
LITERATURE REVIEW

2.1 General Work on Data Quality

Numerous models, evaluation methodologies, and improvement techniques have been developed in the area of Information Quality (IQ) [23, 24, 28, 56, 60]. IQ researchers often regard quality as "fitness for use" [3], so the user's perception of quality and the intended use of the data prevail in these approaches. Wang et al. [60] proposed an attribute-based model to tag data with quality indicators. They suggest a hierarchy of data quality dimensions with four major dimensions: accessibility, interpretability, usefulness, and believability. These dimensions are in turn split into other factors such as availability, relevancy, accuracy, credibility, consistency, completeness, timeliness, and volatility. Mihaila et al. [28] identified four Quality of Data parameters: completeness, recency, frequency of updates, and granularity. Lee et al. [23] distinguished five dimensions of data quality: accessibility, relevancy, timeliness, completeness, and accuracy; each considered a performance goal of the data production process. Lee et al. [24] developed a methodology for IQ assessments and benchmarks called AIMQ. AIMQ is based on a set of intrinsic, contextual, representational, accessibility IQ dimensions, which are important to information consumers. These dimensions were first devised by Strong et al. [56] as categories for high-quality data. Naumann and Rolker [45] proposed an assessment-oriented classification of IQ criteria based on three sources of IQ, namely the user, the source, and the query process. More recently, Naumann and Roth [46] analyzed how well modern (relational) DBMS meet user demands based on a set of IQ criteria. All these works offer valuable contributions for better understanding of data quality problems and challenges, but they fail to provide quantitative measures for the quality dimensions or indicators proposed.

2.2 Work on Data Quality for Cooperative Information Systems

Data Quality has also been studied in the context of Cooperative Information Systems, where more pragmatic approaches have emerged [26, 29, 44, 53]. Mecella et al. [26] describe a service-based framework for managing data quality in cooperative information systems, based on an XML model for representing and exchanging data and data quality. Scannapieco et al. [53] developed the DaQuinCIS architecture and the D2Q (Data and Data Quality) model for managing data quality in cooperative information systems. Naumann et al. [44] presented a model for determining the completeness (i.e., a combination of density and coverage) of a source or combination of sources. Missier et al. [29] defined the notions of quality offer and quality demand within cooperative information systems, and modeled quality profiles as multidimensional date cubes. Bouzeghoub and Peralta [9] analyzed existing definitions and metrics for data freshness in the context of a Data Integration System. All of these works address quality issues that arise in the presence of multiple sources, in particular problems related to data exchange, data integration, and notification services among the sources. In our work, we are primary concerned with the quality of data within a single source, hence most such issues are neither applicable to us nor addressed by our model. Yet we believe our model complements works in Cooperative Information Systems and Data Integration System because they do not typically provide solutions for measuring the inner quality of sources.

2.3 Work on Biological Data Quality

Some work has been proposed in the context of biological data quality. Particularly, the research by Müller et al. [33] identifies the main errors involved in the process of genome data production as well as their corresponding data cleansing challenges. A thorough examination of the quality of the human genome DNA sequence was described by Schmutz et al. [54]. Both focus on assessing the quality of the sequenced data, but our approach is also concerned with the

annotations about the sequenced data. Recently, the management data quality has been studied in the context of the life sciences [48, 30]. Missier et al. [30] proposed the Qurator system, which allows the specification of user's personal quality functions into the so called "quality views". These views are compiled into Web services that can then be embedded within the data processing environment. Preece et al. [48] also describe a framework, based on the Qurator project, for managing information quality in e-Science, using ontologies, semantic annotation of resources, and data bindings. It allows scientists to define the quality characteristics that are of importance in their particular domain, rather than specifying generic, domain-independent quality characteristics. Our approach differs from these works in that it aims to define rather general and objective quality dimensions that can be computed in an automated way, i.e., does not require user's input.

2.4 Other Quality-Related Work

Research efforts in the areas of Quality of Service and Digital Libraries have also explored the characteristics and the role of quality [55, 6, 57, 4]. Quality of Service has mainly been developed to support distributed multimedia applications, which transmit and process audiovisual data streams. Quality of Service comprises the quality specifications, mechanisms, and architecture necessary to ensure that user and/or application requirements are fulfilled [55, 6]. In the context of Digital Libraries, Sumner et al. [57] analyzed the dimensions of educators' perceptions of quality in digital library collections for classroom use. They found that metadata influences how the quality of the collections is perceived. Beall [4] describes the main types of errors in digital libraries, both in metadata and in actual documents; and offers suggestions for managing digital library data quality. The quality problems that have been investigated in the contexts of Quality of Service and Digital Libraries differ from those existing in biological databases.

2.5 Work on Semistructured Data

Finally, works on semistructured data modeling are also relevant to us because such data models have been extensively used in the biological domain, and because our quality model uses an underlying semistructured data model. Most of the models proposed for semistructured data [1, 11, 12, 25] share a common underlying representation, which is either a graph or a tree with labels on the nodes or on the edges. Abiteboul et al. [1] use an edge-labeled graph to represent semistructured data. UnQL and LORE are based on an edge-labeled tree representation [11, 25]. Calvanese et al. [12] use the basic data model for semi-structured data (called BDFS) in which both databases and schemas are represented as graphs. The work by Scannapieco et al. [53] provides a good example of the usage of a semistructured data model (in particular, XML) to represent both data and quality metadata.

CHAPTER 3
QUALITY ESTIMATION MODEL

We present a new model for estimating the quality of biological data in genomics repositories. The model comprises a set of measurable quality dimensions, and a set of quantitative measures that can be systematically computed to provide a score for each quality dimension. Quality dimensions and measures are integrated into a semistructured data model, which is suitable for representing both data and quality metadata, and can accommodate a wide variety of data models.

3.1 Definitions

We define *Data Quality* as a measure of the value of the data. Since value is a rather intangible concept, we decompose it along five different quantifiable dimensions.

Quality dimensions are aspects of the quality of data which either the user or the data provider is interested in measuring. Since we aim for quality dimensions that can be quantified, we need to specify how the quality dimensions will be measured. The particular formula or algorithm by which each dimension is assigned a score is called a *measure*.

We refer to the set of quality dimensions of a data item as its *quality metadata*, and we represent it as a vector where each entry contains the data item's score on a quality dimension, e.g., $Q \equiv [d_1,d_2,\ldots, d_n]$ with $d_1,d_2,\ldots, d_n$ being the scores for the n quality dimensions.

3.2 Quality Dimensions

In order to identify suitable quality dimensions for our model, we looked for dimensions that met the following criteria. First, the dimension could be *objectively measured*, meaning that no subjective appraisal or interpretation was needed to assess a score for the dimension. Second, the dimension could be *efficiently computed*, meaning that the computation of new and updated scores for the dimension should be fast enough to allow its use in real scenarios where the

underlying biological data is constantly being updated. Third, the dimension was biologically relevant, meaning that it effectively captured criteria directly or indirectly used by biologists when assessing the quality of data. The relevancy for biology was preliminary judged by the authors, then validated by field-experts, and lastly confirmed experimentally.

Using the criteria described above a set of seven quality dimensions was selected, namely *Density*, *Freshness*, *Age*, *Stability*, *Uncertainty*, *Linkage*, and *Redundancy*. The first five of these dimensions are *per*-record dimensions and the last two are *cross*-record dimension. Per-record dimensions are dimensions that consider quality aspects of a *single* record solely (i.e., they assess records on an individual basis). Cross-record dimensions are dimensions that consider quality aspects across a set of records *collectively* (i.e., they asses the interactions among records).

Next we provide an informal description of the selected quality dimensions, which will be formalized later when we present their measures. Hereafter, we use the term "*data item*" to denote either logical data units with structure such as records or fields of a record, or the value of any such logical data unit.

3.2.1 Per-Record Dimensions

3.2.1.1 Density

Density is a spatial dimension that provides an assessment of the amount of information conveyed by a data item *d*. The amount of information can be measured as the number of (possibly nested) data items within the data item *d*. This recursive definition takes a concrete and natural form under the chosen data model. The intuition behind this dimension is as follows. A data item which consists of many other data items is considered "dense" because much information is comprised in it, hence its density score would be high. On the other hand, a data item which consists of just a few other data items is deemed as "light" since little information is comprised in it, and therefore its density score would be low.

Initially, Density was deemed sufficient for capturing the notion of "amount of information" in a biological record, but later we found that two other aspects related to density were also relevant to the biologists, namely the feature and publication information. By feature information we refer to the features annotated in the Feature Table [21] of a genomic record, which may describe genes, gene products, and regions of biological significance in the sequence. By publication information we refer to the references contained in a genomic record, which may be to publications in a journal article, book chapter, book, thesis, monograph, proceedings chapter, proceedings from a meeting, or patent. These two information dimensions, named *Features* and *Publications*, were incorporated into our model as sub-dimensions of Density because despite being encompassed by the Density dimension, feedback gathered from biologists suggested that they had significant biological meaning as to justify its separate treatment.

Both Features and Publications dimensions can be measured by counting the number of feature key and reference elements, respectively. The intuition for these new dimensions is that the more feature keys and the more references a record has, the higher its features and publications scores are.

3.2.1.2 Freshness

Freshness is a temporal dimension that indicates how up to date the contents of a data item *d* are. It can be measured as a function of the time elapsed since the last update of the data item *d*, using an exponential decay. The intuition behind the Freshness dimension is as follows. If a data item has recently been updated, it is considered to be "up to date", and its freshness score would therefore be high. Conversely, if a data item has remained unchanged for a long period of time it is considered "outdated" and its freshness score would then be low.

3.2.1.3 Age

The Age dimension indicates how old the contents of a data item *d* are, so it is a temporal dimension, too. It can be measured as a function of the time elapsed since the creation of the data item *d*, using an exponential decay. The intuition behind this dimension is as follows. If a data item was created long time ago it is considered "old" and its age score would therefore be high. Conversely, if a data item has been recently created, it is considered "young", and its age score would hence be low.

3.2.1.4 Stability

The Stability dimension captures information about changes in the contents of a data item *d*, which can be obtained from the version history available in main biological repositories. This is both a temporal and a provenance dimension since it keeps track of changes applied to the data though time. Stability can be measured as the magnitude of the changes undergone by the data item relative to its size, and weighted by a function of the time elapsed since the change occurred. This weighting function diminishes the influence of older updates in favor of recent ones. The intuition behind the Stability dimension is the following. If a data item has not undergone large changes during a recent period of time, it is regarded as "stable" and its stability score would therefore be high. Conversely, if a data item has undergone considerably large changes recently, it is considered "unstable" and its stability score would then be low.

3.2.1.5 Uncertainty

The Uncertainty dimension is an indicator of the lack of evidence for the contents of a data item *d*. In the biological context, this typically refers to ambiguities associated to the particular experimental procedure or method used to obtain the data. More specifically, in our target databases the ambiguities or uncertainties come primarily from the sequence data, and are expressed as degenerate bases. The Uncertainty dimension can hence be measured as the fraction

of ambiguous values (i.e., degenerate bases), relative to the total length of the sequence. The intuition behind the Uncertainty dimension is as follows. If a data item contains many ambiguous values, we regard it as an "uncertain" item, and its uncertainty score would therefore be high. On the contrary, if a data item has no ambiguous values, we would consider it "certain" and its uncertainty score would then be low.

3.2.2 Cross-Record Dimensions

3.2.2.1 Linkage

Linkage is a spatial dimension that provides information about the incoming and outgoing links of a data item *d* (a record, in this context). In biological databases, records can be linked to other relevant records, published articles, etc. Such information is typically represented as an interaction graph that consists of a set of nodes representing records, and a set of directed edges or links between nodes representing relationships between records. For example, a link between a record in the RefSeq database [39, 49] and an entry in the PubMed database [43,61] indicates that the corresponding PubMed article describes or uses the information in the RefSeq record.

Initially, we considered Linkage as a single dimension, but after examining the rich information offered by different link types within the NCBI databases, we decided to split the Linkage dimension into four mutually exclusive sub-dimensions, namely *Literature Links*, *Gene Links*, *Structure Links*, and *Other Links*. The Literature Links dimension comprises links to or from literature databases, specifically NCBI's PubMed, NCBI's Online Mendelian Inheritance in Man (OMIM), and NCBI's Online Mendelian Inheritance in Animals (OMIA). The Gene Links dimension accounts for links to or from gene and genome databases, in particular NCBI's Gene, NCBI's HomoloGene, and NCBI's Genomes. The Structure Links dimension contains links to or from structure and domain databases, specifically NCBI's Structure (MMDB), NCBI's 3D Domains, and NCBI's Conserved Domains (CDD). Lastly, the Other Links dimension covers all

other links not included in any of the previous dimensions; for example, links to related sequences across databases. This division, which we believe represents important but different biological characteristics of the genomic data, was refined with the help of expert collaborators at our university.

All of the linkage dimensions can be measured as a link count over the respective target databases. The intuition behind the Linkage dimension (and sub-dimensions) is that a data item with many links in its interaction graph would have a high linkage score whereas one with few links would have a low linkage score.

3.2.2.2 Redundancy

Redundancy is a spatial dimension that captures information about the number of redundant data items (records, in this context) with respect to a data item *d* (a record, too). In biological databases, two records are considered redundant (with respect to each other) if their sequence similarity is significantly high. Annotations about the sequence data could also be incorporated into a general notion of redundancy (which was our initial approach), but the problem with this approach is that measuring the redundancy at the annotations level would lead to expensive string comparisons among records in the database, which we cannot afford. Even measuring the redundancy at the sequence level would be computationally costly if no extra information is provided by the data source. Luckily, our target database runs BLAST over the stored records periodically, and pre-computes the "neighbors" or related sequences for every record. Using this information, the Redundancy dimension of a data item *d* can be measured as the number of neighbors of *d*. The intuition behind this dimension is that a data item with many neighbors would have a high redundancy score whereas one with few neighbors would have a low redundancy score.

3.2.3 Possible Extensions

It is worth mentioning that the proposed set of quality dimensions (described above) could be extended to include other aspects of quality that may also be relevant in the biological context. The selection of an appropriate set of dimensions is, in fact, one of the biggest challenges in the area of data quality, and in our context this decision is influenced by what information is currently offered at the data sources. As an example, we next describe two quality dimensions that we intended to include but decided to drop because we did not have enough information to effectively measure them yet. When such information becomes available, our model can be extended to accommodate these (and other) new dimensions.

3.2.3.1 Query pattern

This dimension would capture information about the query pattern of a data item (e.g., how many times the data item has been queried, when did those queries occurred, etc). Such query information could be used similarly to the way in which the update history is used in the Stability dimension. The intuition behind the query pattern dimension of a data item would be as follows. If the data item has been frequently queried (i.e., used by scientists), its score would be high based on the assumption that scientists normally use data they trust to be correct. On the other hand, if no queries are issued to the data item for a long period of time, its score would be low.

Although this dimension seemed useful according to our predefined criteria, we were not able to include it in our final set of dimensions because the database we were using did not provide information about the queries made by users.

3.2.3.2 Correctness

This dimension would indicate the correctness (or accuracy) of a data item. Although measuring the correctness of a biological data item is not a trivial task, we could reduce the scope of this dimension to evaluating the sequence data of a biological record, which indeed is a

key aspect for biologists. If we restrict the Correctness dimension in this way, we could then use the quality scores (i.e., base-calling scores) stored in our target database to compute a score for this dimension. The problem was that such quality scores were not consistently available for all the data stored in the database, hence we decided not to include it in our final set of dimensions.

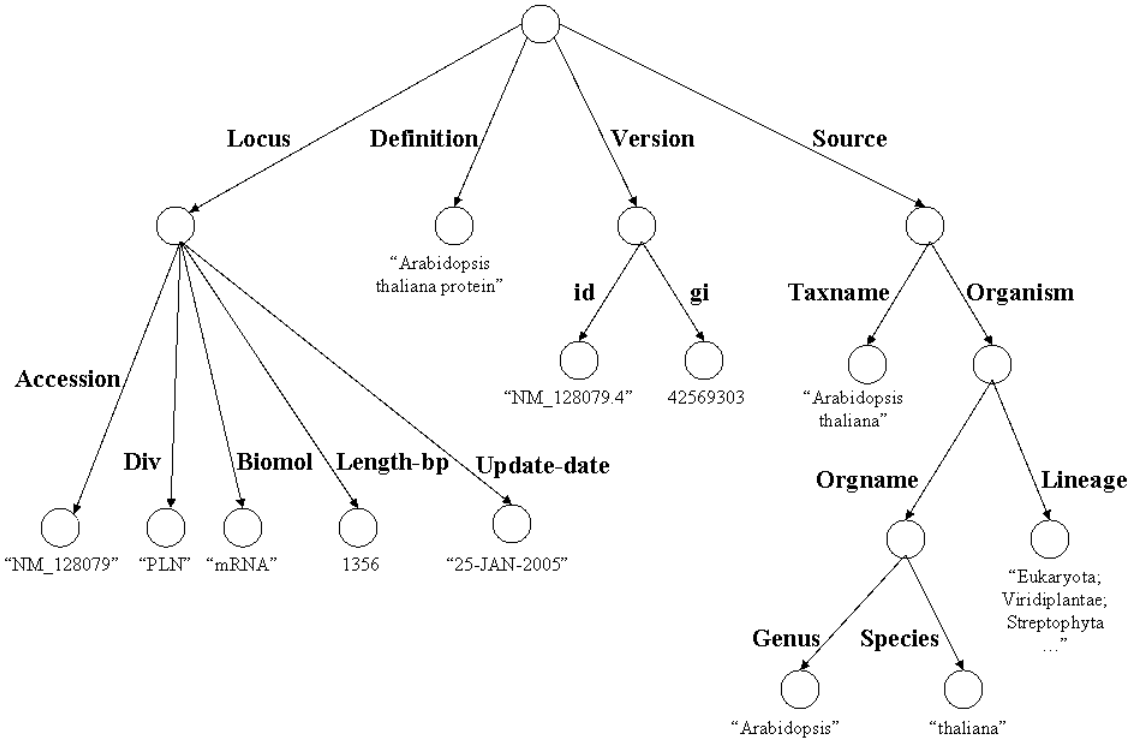
3.3 Measures for the Quality Dimensions

3.3.1 Underlying Data Model

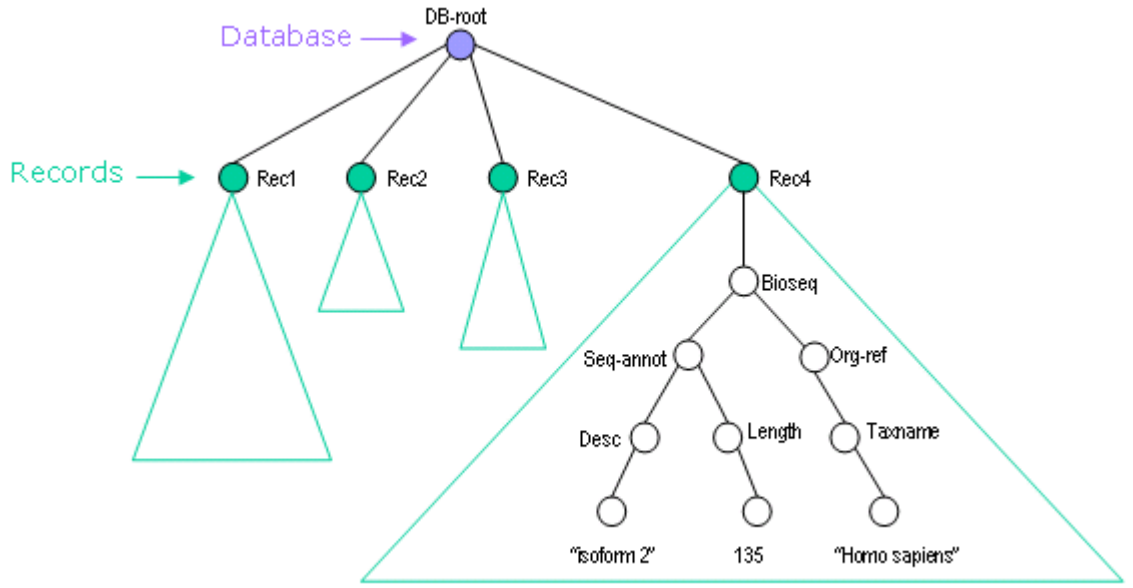
Before we can formulate measures for the quality dimensions, we need to select a data model in which the underlying biological data is to be represented. In this work, we chose the semistructured data model. Semistructured data is commonly described as "schemaless" or "self-describing" data [1, 10] because the schema of the data is contained within the data. A semistructured data model generally represents data hierarchically (i.e., in a tree-like structure[2]), with actual data represented at the leaf nodes and schema information encoded in upper layers of the hierarchy (i.e., internal nodes). Here, leaf nodes store *atomic* data items or values, which can be either strings or numbers. Internal nodes represent *complex* data items, which are collections of other data items. The Abstract Syntax Notation number One (ASN.1) [18] and the Extensible Markup Language (XML) [62] are two examples of semistructured data models.

Figure 3-1A shows a fragment of a genomic record using the semistructured model proposed by Abiteboul et al. [1], and Figure 3-1B sketches a semistructured representation of a database, where the root of the tree represents the entire database, nodes immediately below the root represent records in the database, and nodes below them represent data items within records. It is worth noting that our quality estimation model can be used with any semistructured data model that adheres to the principles stated above, hence our conceptual data model is very

[2] Strictly speaking, the semistructured data model allows for cycles in the data, hence a graph representation should be used instead of a tree. However, when the nature of the data is acyclic, a tree-like structure can be assumed.



A



B

Figure 3-1. Examples of semistructured representations of data. A) Fraction of a genomic record represented in the semistructured model by Abiteboul et al. B) Sketch of a semistructured representation of an example database with four records.

generic. When we discuss the implementation of the quality model, we will specify the format and syntax of the particular data model chosen.

Several reasons justify the selection of a semistructured data model in this context. First, a vast amount of biological data is currently available as semistructured data (e.g., GenBank [35, 5], EMBL [17, 22], and DDBJ [42] publish their data in XML). Second, semistructured models have proven useful at representing biological data and its intrinsic complexities, which is demonstrated by the increasing number of XML-based languages developed for biology (e.g., BioML, BSML, AGAVE, GeneXML). Third, a semistructured data model can seamlessly represent both data and metatada, which is a desirable feature in our quality model. Finally, it can accommodate a variety of other data models, thus making it possible to estimate the quality of a wide variety of repositories that use different data representations. Using the semistructured data model allows us to measure the different quality dimensions in a bottom-up fashion, which is described next.

3.3.2 Measures

3.3.2.1 Density measure

Our informal previous description of the density measure involved counting the number of (possibly nested) data items for a given data item. Now we refine such definition. Under the adopted hierarchically-structured data model, a complex data item is a collection of other data items (i.e., it contains nested data items), which can be either complex or atomic data items. Also under this model, an atomic data item cannot contain nested data items; it can only have data values like strings or numbers. Hence, the recursive definition of counting data items within data items becomes natural when we apply it to complex and atomic data items of our data model. Recursion thus stops at the atomic data items.

The density score of an atomic data item d is defined as 1, for any d. This means that each atomic data item, regardless of its size, has the same contribution to the total amount of information of a record. For example, if the value of the atomic data item d_1 in the data tree is a large string s_1, and the value of the atomic data item d_2 is a short string s_2, then both d_1 and d_2 will have a density score of 1, meaning that each represents one unit of information.

Once we know the density score for atomic data items at the bottom level of the tree, we can recursively compute the density score for complex data items in upper levels of the tree. The measure for the Density dimension of a complex data item d is given in Equation 3-1, where n is the number of components (i.e., direct descendants) of d, and D_i is the density score of the ith component of d in the data tree. This measure is equivalent to the size (i.e., number of nodes) of the subtree whose root node is d.

$$D = 1 + \sum_{i=1}^{n} D_i \,. \tag{3-1}$$

The density score can take on values from the interval [1, ∞[, where 1 represents the minimum density value, and there is no upper limit on the density value.

Measures for sub-dimensions. We provide here the measures for the *Features* and *Publications* sub-dimensions of Density. These measures differ from the density measure in two ways: first, they do not count nested data items; second, they only apply to certain complex data items of interest (hence, we do not provide a measure for atomic data items in this case). Given a complex data item d representing a record r, the Features score T of item d is defined as the number of complex data items representing feature keys in r, and the Publications score P of d is defined as the number of complex data items representing references in r. Each complex data item representing either a feature key or a reference thus contributes with one unit to the corresponding measure.

3.3.2.2 Freshness measure

We previously described the freshness measured as a function of the time elapsed since the last update of the data item *d*, using an exponential decay. This notion is formalized in Equation 3-2, which specifies how to obtain the freshness score for an atomic data item *d*.

$$F = e^{-\sigma\left(\frac{t-u}{f}\right)}. \tag{3-2}$$

In Equation 3-2 *t* is the current time, *u* is the time when *d* was last updated, *f* is the frequency of update of the database (represented in the same time units as the subtraction in the numerator), and σ is a parameter that controls the decay rate of the freshness score. The role of *f* is to scale the time elapsed since last update to units that reflect the rate at which the database gets updated. The exponential decay gives more weight to recent past than to distant past, and also ensures that the freshness score takes values between 0 and 1.

For a complex data item *d*, the freshness score is defined as the average of the freshness scores of its components (i.e., direct descendants of *d* in the data tree).

The freshness score can take on values from the interval [0, 1], where 0 and 1 denote the minimum and maximum freshness values, respectively.

3.3.2.3 Age measure

The age measure was previously described as a function of the time elapsed since the creation of the data item *d*. This notion is formalized in Equation 3-3, which specifies the measure for the Age dimension of an atomic data item *d*.

$$A = 1 - e^{-\beta\left(\frac{t-c}{f}\right)}. \tag{3-3}$$

In Equation 3-3 *t* is the current time, *c* is the time when *d* was created, *f* is the frequency of update of the database (represented in the same time units as the subtraction in the numerator),

and β is a parameter that controls the decay rate of the age score. The role of f is to scale the time elapsed since creation to units that are in accordance to the database update rate. The transformation applied to this scaled time produces large increases in age at the beginning and then slows down as time passes by. This also ensures that the age score is between 0 and 1.

For a complex data item d, the age score is defined as the average over the age score of its components (i.e., direct descendants of d in the data tree).

The age score can take on values from the interval [0, 1], where 0 and 1 denote the minimum and maximum age values, respectively.

3.3.2.4 Stability measure

In our informal description of the stability measure from previous section we suggested to quantify the magnitude of the updates applied to a data item, and use a time-dependent weighting function to reduce the effect of older updates. We formalize this notion in Equation 3-4, which specifies the measure for the Stability dimension of an atomic data item d.

$$S = 1 - \sum_{i=1}^{n} \Delta(d(i-1), d(i)) \int_{t_i}^{t_{i-1}} \lambda e^{-\lambda t} dt \ . \tag{3-4}$$

In Equation 3-4 n is the number of intervals at which we measure the stability of d, t_i is the time elapsed since the ith interval (with $t_0 \equiv \infty$), $d(i)$ is the state[3] of d at interval i, and $\lambda > 0$ is a free parameter. The function Δ measures the fraction of d that changed between two consecutive intervals. The integral of the exponential function applies a time-decaying weight to the changes undergone by d, effectively giving more weight to recent changes than to old ones. Note that S is initially 0 since $\Delta(d(0), d(1)) = 1$ for any data item d (the default type of any data item at time t_0 is null, and $\Delta(null, d(1)) = 1$ for any $d(1) \neq null$, so the integral evaluates to 1).

[3] The state of a data item is defined by type and contents.

It is possible to express the stability measure from Equation 3-1 in an incremental way (Equations 3-5 and 3-6). The stability score *S* of a data item *d* can be obtained from the *instability I* of *d* score as in Equation 3-5, where the instability score *I* of *d* at time t_k is given by Equation 3-3, and *I* at time t_0 is define to be 1. Equation 3-6 shows that the instability score at time t_k is determined only by the score at time $t_{k\text{-}1}$, meaning that we only need to know what the previous score was (rather than all previous scores since t_0). Note that Equation 3-6 is an exponential moving average with memory depth $e^{-\lambda t_{k-1}}$. The stability measure can therefore be computed incrementally.

$$S_{t_k} = 1 - I_{t_k}. \tag{3-5}$$

$$I_{t_k} = e^{-\lambda t_{k-1}} I_{t_{k-1}} + \left(1 - e^{-\lambda t_{k-1}}\right) \Delta\left(d(k-1), d(k)\right). \tag{3-6}$$

The function $\Delta(d_1, d_2)$ for atomic data items d_1 and d_2 is defined in Equation 3-7. Note that $0 \leq \Delta(d_1, d_2) \leq 1$ for any pair (d_1, d_2). If d_1 and d_2 are numbers, this formula assumes that they are positive. It is also possible to use an approximation to the Edit Distance function if efficiency is a major concern.

$$\Delta(d_1, d_2) = \begin{cases} \dfrac{\text{editDist}(d_1, d_2)}{\max\{\text{length}(d_1), \text{length}(d_2)\}} & \text{if } d_1, d_2 \text{ are strings} \\ \dfrac{|d_1, d_2|}{\max\{d_1, d_2\}} & \text{if } d_1, d_2 \text{ are numbers.} \\ 1 & \text{otherwise} \end{cases} \tag{3-7}$$

The stability *S* of a complex data item *d* is defined as the average over the stability score of its components (i.e., direct descendants of *d* in the tree).

The stability score can only assume values from the range [0,1], where 0 represents minimum stability and 1 represents maximum stability.

3.3.2.5 Uncertainty measure

In the previous section, we described the uncertainty measure as the fraction of ambiguous values relative to the total length of the record's sequence. Since this measure is based on the sequence information, the uncertainty score is only computed for the atomic data item *d* representing the sequence data (i.e., string containing the nucleotide or amino acid sequence); other data items (complex and atomic) would have *null* uncertainty score by default. Equation 3-8 defines the uncertainty measure for an atomic data item *d*.

$$U = \frac{\text{degenerateCount}(d)}{\text{length}(d)}. \tag{3-8}$$

In Equation 3-8, degenerateCount(*d*) is a function that counts the total number of degenerate bases in *d* (assuming that *d* is the atomic data item containing the sequence string), and length(*d*) is the size of the string represented by *d* (i.e., the total number of bases in the sequence). Table 3-1 lists the ambiguity codes for degenerate bases in nucleotide sequences, and Table 3-2 shows the ambiguity codes for degenerate bases in amino acid (i.e., protein) sequences.

Table 3-1. Ambiguity codes for nucleotide sequences.

Code	Meaning (Base)
R	purine (G or A)
Y	pyrimidine (T or C)
K	keto (G or T)
M	amino (A or C)
S	strong (G or C)
W	weak (A or T)
B	not A (G or T or C)
D	not C G or A or T)
H	not G (A or C or T)
V	not T (G or C or A)
N	any base (A or G or C or T)

Table 3-2. Ambiguity codes for amino acid sequences.

Code	Residue
B	aspartate or asparagine
Z	glutamate or glutamine
X	any residue

The uncertainty score can only take values in the range [0,1], where 0 and 1 represent minimum and maximum uncertainty, respectively.

3.3.2.6 Linkage measure

Since Linkage dimension is a cross-record dimension, it is relevant only to data items that represent records; hence we focus only on the measure for complex data item representing records. Previously we mentioned that the four linkage sub-dimensions could be measured as the link count over the target databases encompassed by each sub-dimension; thus there really is just one linkage measure (i.e., link count), which is applied over different databases for each sub-dimension.

Measures for sub-dimensions. Given a complex data item *d* representing a record *r*, the Literature-Links score *L* of item *d* is defined as the number of links from *r* to entries in NCBI's literature databases (PubMed, OMIM, OMIA), the Gene-Links score *G* of *d* is defined as the number of links from *r* to entries in NCBI's gene-related databases (Gene, HomoloGene, Genomes), the Structure-Links score *C* of *d* is defined as the number of links from *r* to entries in NCBI's structure-related databases (MMDB, 3D Domains, CDD), and the Other-Links score *O* of *d* is defined as the number of links from *r* to entries in all other NCBI databases. Each link thus contributes with one unit to the link count. In the context of our target biological databases all links are two-way links; hence the former linkage scores effectively reflect both the number of outgoing and incoming links to/from record *r*.

The score for each linkage dimension can take on values from the interval [0, ∞[, where 0 means that no link exists. There is no upper limit on the value of the linkage score.

3.3.2.7 Redundancy measure

Previously, the redundancy measure was described as the number of "neighbors" of a data item representing a record. Given that Redundancy is a cross-record dimension, we focus only on the measure for complex data items that represent records.

The Redundancy score *R* of a complex data item *d* representing a record *r* is defined as the number of links from *r* to distinct entries in the same database of *r* (e.g., nucleotide-nucleotide or protein-protein relationships). Each of these links thus contributes one unit to the total link count. This link count effectively counts the neighbors of *r* (i.e., redundant data items of *d*).

The score for the Redundancy dimension can take on values from the interval [0, ∞[, where 0 means that no redundant data items exist. There is no upper limit on the value of the redundancy score.

3.3.3 Complexity Analysis of the Measures

Based upon the chosen semistructured data model, we define *n* as number of nodes in the tree representing a record *r* (see Figure 3-1B), *l* as the number of leaf nodes in the tree, *c* as the number of child nodes (i.e., direct descendants) of an internal node of the tree, *d* as the length of the largest data string stored at a leaf node, and *s* as the length of the sequence string of a record *r*. Also, let l_l, l_g, l_s, l_o, and l_n be the number of literature links, gene links, structure links, other links, and neighbor links of record *r*, respectively. We first present the complexity analysis for the measures of the per-record dimensions and then for the measures of the cross-record dimensions. Our analysis distinguishes between atomic and complex data items; as well as between initialization and update times. Initialization time refers to the time when new biological data (usually in the form of a record) is added to the database, so the scores of the quality

Table 3-3. Time complexity of the per-record measures.

Type of data item	Measure	Initialization	Update
Atomic	Uncertainty	$O(s)$	$O(s)$
	Stability	$O(1)$	$O(d^2)$
	All others	$O(1)$	$O(1)$
Complex	Uncertainty	$O(1)$	$O(1)$
	All others	$O(c)$	$O(c)$

Table 3-4. Time complexity of the cross-record measures.

Type of data item	Measure	Initialization	Update
Complex	Redundancy	$O(l_n)$	$O(l_n)$
	Literature Links	$O(l_l)$	$O(l_l)$
	Gene Links	$O(l_g)$	$O(l_g)$
	Structure Links	$O(l_s)$	$O(l_s)$
	Other Links	$O(l_o)$	$O(l_o)$

dimensions should be given an initial value. Update time refers to the time when the biological data is updated (parts of a record are modified), so the score of each quality dimension needs to be updated to reflect the change in the underlying data. The main results obtained from the following analysis are shown in Tables 3-3 and 3-4.

3.3.3.1 Complexity of the per-record measures

Initialization Time. At initialization time, any of the per-record measures except Uncertainty can be computed in constant time i.e., $O(1)$ for an *atomic* data item. The Uncertainty measure takes $O(s)$ time, since the entire sequence string of a record needs to be scanned at initialization time.

On the other hand, computing the initial per-record measures for a *complex* data item takes time proportional to the number of child nodes of the complex data item, i.e., $O(c)$.

For a given record r, initializing the scores of the per-record dimensions can be done in $O(n + s)$ time where n results from a post-order traversal of the tree representing r (n is the aggregate of c over all internal nodes), and s results from scanning the sequence string of the

record, which can be significantly large for some records (especially those that represent complete genomes).

Update Time. At update time, any of the per-record measures except Stability and Uncertainty can be computed in constant time for *atomic* data items. Regarding Stability, its worst-case complexity occurs when the type of the atomic data item is string, since the function Δ computes the edit distance between the old and new values of the string. If d_1 and d_2 are the lengths of the old and new strings, respectively, then computing the edit distance takes $O(d_1 * d_2)$, and the worst case complexity for the Stability measure becomes $O(d^2)$. On the other hand, the complexity for the Uncertainty measure at update time is $O(s)$.

Updating the per-record scores of a *complex* data item merely involves re-computing an average or similar aggregate over the complex item's child nodes, which requires $O(c)$ time. The only exception is Uncertainty, which is defined to be *null* for all complex data items, hence it takes $O(1)$ time.

For a given record r and its previous version r', updating the scores of the per-record dimensions can be done in $O(r.n + \max\{r.d, r'.d\}^2 * r.l)$ time where $r.n$ results from a post-order traversal of the tree representing r (n is the aggregate of c over all internal nodes), and $\max\{r.d, r'.d\}^2 * r.l$ results from updating the stability score for all the strings at the leaves of the tree.

3.3.3.2 Complexity of the cross-record measures

The time complexity analysis in this section concerns only *complex* data items that represent records, since we are dealing with measures across records.

Initialization Time. At initialization time, the Redundancy measure for a given record r takes time proportional to the number of neighbors of r, i.e., $O(l_n)$. Likewise, for a given record r the measure for the Literature-Links dimension takes time proportional to r's number of links

to/from literature databases i.e., $O(l_l)$; the Gene-Links dimension takes time proportional to r's number of links to/from gene databases, which is $O(l_g)$; the Structure-Links dimension takes time proportional to r's number of links to/from structure databases, which is $O(l_s)$; and the Other-Links dimension takes time proportional to r's number of links to/from other databases, which is $O(l_o)$.

Hence, initializing the scores of all the cross-record dimensions for a given record r can be done in $O(l_n + l_l + l_g + l_s + l_o)$ time.

Update Time. At update time, the complexity of the three cross-record measures is the same as for initialization time. Hence, updating the scores of the cross-record dimensions for a given record r can be done in $O(l_n + l_l + l_g + l_s + l_o)$ time.

3.4 Quality-Aware Operations

Since we are primarily concerned with biological data, we must consider a scenario where data is constantly being updated and queried. Thus, we need to address the issues of how the quality measures described above are affected by data manipulation operations (e.g., insertions, deletions, and updates of fields or records), and how the result of these operations is extended to include the quality scores. For this purpose, we consider a core set of operations over hierarchically-structured data. Such set includes query operations such as selection, and maintenance operations such as insertion, update, and deletion.

For the subsequent discussion, let $v_1,v_2,\ldots,v_k$ (with $v_k = v$) be the sequence of adjacent vertices (or nodes) from the root of the tree to the node of interest v. We refer to $v_1,v_2,\ldots,v_k$ as the "path" of v in the tree, and to the set $\{v_1,v_2,\ldots,v_{k-1}\}$ as the "ancestors" of v in the tree.

3.4.1 Query Operations

We consider selection queries here. In the context of hierarchically structured data, a *select* operation consists of navigating a path given by the user and then returning all or part of the contents of the node located at the end of the path.

3.4.1.1 Selecting a node and returning its contents

The Select operation takes as input parameter a path $p = v_1, v_2, \ldots, v_k$. The Select operator navigates path p until it reaches its last node v_k, and returns the contents of this node. If v_k is a leaf node, its contents refers to the atomic data item (i.e., data value) stored at v_k. If v_k is an internal node, its contents refer to the subtree rooted at v_k. Optionally, the Select operator can take a second input parameter *filter*, which specifies which of the quality dimensions to retrieve along with the data. If such argument is omitted, the default behavior is to retrieve all the quality dimensions.

Under a select operation, the quality scores of the target node (v_k in this case) will not be affected since this is a 'read' operation (i.e., no changes are made to the contents of the node).

This operation returns both the contents and quality metadata of v_k (i.e., scores for the requested quality dimensions).

3.4.2 Maintenance Operations

We consider three types of maintenance operations: inserts, deletes, and updates. In the context of hierarchically structured data, each of these operations navigates an input path, and perform the corresponding insertion, deletion or update to the last node of the path. Figure 3-2 illustrates an update operation (other operations work in a similar way): the state of the database at times t_i and t_{i+1} (when the update occurs) is shown. A leaf node (colored in red) is updated at time t_{i+1}, which causes its quality metadata to be updated. Then the quality metadata of the leaf's ancestors is also updated (colored in blue).

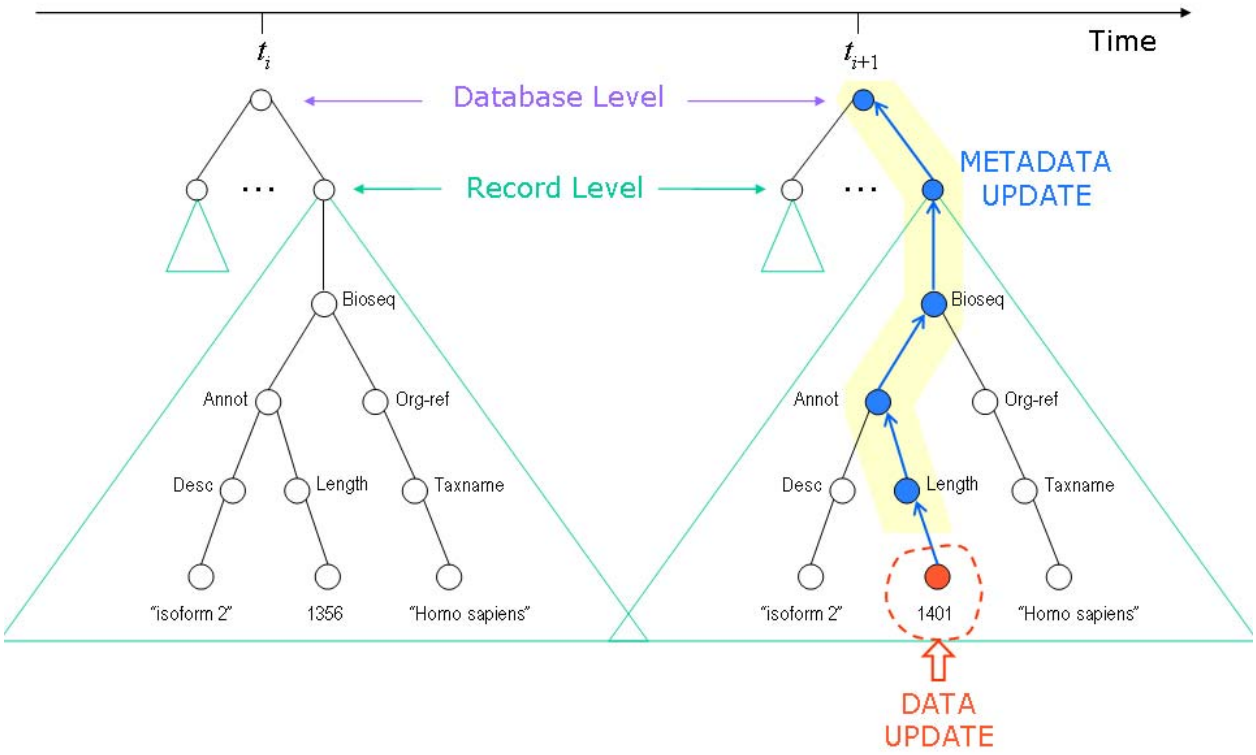


Figure 3-2. Illustration of an update operation. The data stored at a leaf node (shown in red) is updated at time t_{i+1}, causing the quality metadata of the leaf and its ancestors to be updated (changes are propagated bottom-up).

3.4.2.1 Inserting a node

The Insert operation takes two input parameters: the node v to be inserted and the path $p = v_1, v_2, \ldots, v_k$ at the end of which node v_k will be inserted. The Insert operator navigates path p until it reaches its last node v_k, and inserts node v as a child of v_k. Since v is a new node, the scores of all its quality dimensions need to be initialized. Initial scores are specified in the previous Measures section. Next we sketch the steps involved in computing the quality scores under a data insertion.

- **Step 1**. If v is a single node (i.e., an atomic data item), compute v's quality scores as described in the Measures Section.
- **Step 2**. If v is the root of a subtree (i.e., a complex data item), recursively compute the quality scores for all descendants of v and then for v, as described in the Measures Section.
- **Step 3**. Add v as a child node of v_k.
- **Step 4**. Propagate the effect of this insertion to the ancestors of v so that their quality scores get updated, too.

This operation returns both the path to the recently inserted node and the quality metadata associated to this new node.

3.4.2.2 Deleting a node

The Delete operation takes as input parameter a path $p = v_1, v_2, \ldots, v_k$. The Delete operator navigates path p to its last node v_k, and deletes this node. When the node v_k is deleted from the hierarchical data model, the quality scores of v_k's parent (v_{k-1}) need to be recomputed to reflect the deletion. Next we sketch the steps involved in updating the quality scores under a data deletion.

- **Step 1**. If v_k is a leaf node (atomic data item), delete v_k.
- **Step 2**. If v_k is an internal node (complex data item), we need to distinguish two cases: A) single node deletion, and B) subtree deletion.
- **Step 2A**. Single node deletion case: Move v_k's child nodes to path $v_1, v_2, \ldots, v_{k-1}$ so that they become direct descendants of v_{k-1}, and keep their quality scores unchanged. Then delete v_k.
- **Step 2B**. Subtree deletion case: Delete v_k and all its descendants.
- **Step 3**. Update the quality scores of node v_{k-1} and propagate this update to all its ancestors, as described in the Measures Section.

This operation returns the updated quality metadata of the deleted node's parent, v_{k-1}.

3.4.2.3 Updating a node

The Update operation takes two input parameters: a node v_k^{new} which is the updated version of node v_k, and a path $p = v_1, v_2, \ldots, v_k$. The Update operator navigates path p to its last

node v_k, and updates this node with its newer version, v_k^{new}. The scores of the quality dimensions of v_k then need to be updated to reflect the change in the contents of v_k. Next we sketch the steps involved in updating the quality scores under a data update.

- **Step 1**. If v_k is a leaf node (atomic data item), update v_k according to v_k^{new}, and recompute the quality scores of v_k as described in the Measures section.
- **Step 2**. If v_k is an internal node (complex data item), update v_k according to v_k^{new}, and find a correspondence $F(A,B)$ between the set A of direct descendants of v_k^{new} and the set B of direct descendants of v_k. Then, for all pairs of nodes $\langle c_i,c_j \rangle$ in F where c_i is a child of v_k^{new} and c_j is a child of v_k, recursively call the Update operation with parameters c_i and $path(c_j)$. For all child nodes c_i of v_k^{new} that do not map to child nodes in v_k, call the Insert operation with parameters c_i and $path(c_i)$. For all child nodes c_j of v_k that do not map to child nodes in v_k^{new}, call the Delete operation with parameters c_j and $path(c_j)$.
- **Step 3**. Propagate the effect of the update to all ancestors of v_k, as described in the Measures section.

  This operation returns the updated quality metadata of the updated node.

3.4.3 Complexity Analysis of the Operations

We now provide the complexity analysis of the quality-aware operations described previously. Let $p = v_1,v_2,\ldots,v_k$ be the path to a node v in the data tree with $v = v_k$, and p_l be the length of path p measured as the number of nodes along the path ($p_l = k$). Also, let n be the total number of nodes in the tree, c_{max} be the maximum number of children nodes of a node in the tree, and $t_{read}(v)$ be the time required to read the contents (data values) of node v. We reuse some of the definitions from the Complexity Analysis of the Measures section, in particular: l (the number of leaf nodes in the tree), c (the number of child nodes of an internal node of the tree), d (the length of the largest data string stored at a leaf node), s (the length of the sequence string of record r), l_l (the number of literature links of r), l_g (the number of gene links of r), l_s (the number of structure links of r), l_o (the number of other links of r), and l_n (the number of neighbor links of r). For the following analysis, we assume that v is the node on which the operations will be performed, and its path is denoted by p.

3.4.3.1 Complexity of the query operations

Select. The Select operation takes time $O(p_l + t_{read}(v))$ since we need to traverse the path p to reach node v and read its contents. Our analysis does not include the time required to actually search for node v (i.e., find its path) in the tree because we assume that all operations are given the path of v as input and therefore v is found in $O(p_l)$. Consequently, v has to be searched for (and its path found) before any operation can be invoked. Since this is a preprocessing step common to all our operations but is not part of the operations themselves (as defined here), we intentionally exclude its time complexity, which is O(n), from the analysis.

3.4.3.2 Complexity of the maintenance operations

The time needed to perform each Insert, Update, and Delete operation is $nav + op + prop$, where nav is the time needed to navigate through path p to node v, op is the time needed to perform a given operation (e.g., insert, update, or delete) on node v, and $prop$ is the time needed to propagate the changes up the tree to the ancestors of v (see Figure 3-2 for an illustration of a maintenance operation). The nav time is $O(p_l)$ for all operations. The $prop$ time is $O(p_l * c_{max})$ for all operations. The op time is analyzed next.

Insertion of *v*. The time to perform an insert operation at node v depends on whether we insert a single node or a subtree. If a single node is inserted and this node is not the sequence node, insertion time is $O(1)$ since the initialization of the quality measures of an atomic data item can be done in constant time (see the Complexity of the Measures Section). However, if the node being inserted is the sequence node (i.e., the atomic data item representing the sequence data), insertion time is $O(s)$ because the uncertainty measure needs to be computed in this case. If on the other hand a subtree s is inserted, the insertion time is $O(n_s)$ where n_s is the number of nodes in s. This is because we have to initialize the quality measures of each node within the s subtree. Moreover, if s contains the sequence node, an additional cost of $O(s)$ is incurred. In summary, for

insertions not involving the sequence node, *op* is $O(1)$ when a single node is inserted, and $O(n_s)$ when a subtree is inserted. For insertions involving the sequence node, *op* is $O(a_s)$ when a single node is inserted, and $O(n_s + s)$ when a subtree is inserted. In the special case when the inserted subtree represents a record, *op* is $O(n+s+l_n+l_l+l_g+l_s+l_o)$.

Deletion of *v*. The time to perform a delete operation at node *v* is constant since we do not update the quality metadata of *v* or any of its descendants; hence most of the time taken by this operation is spent in the propagation phase. For deletions, thus, *op* is $O(1)$.

Update of *v*. The time to perform an update operation at node *v* depends on whether *v* is a leaf node or internal node. If *v* is a leaf node, the update time is $O(d_{new} * d_{old})$ with d_{new} and d_{old} being the newer and older contents of *v*, which results from the worst-case time for the stability measure (see the Complexity of the Measures section). Instead, if *v* is a subtree *s*, the update time is $O(c_{max}' * c_{max} * h_s)$ where h_s is the height of subtree *s*, and c_{max}' and c_{max} are the newer and older values of the maximum number of child nodes for an internal node in *s*, respectively. Such time complexity is driven by our heuristic algorithm[4] for finding a correspondence between the direct descendants of *s'* (where *s'* denotes the updated subtree) and the direct descendants of *s*. Moreover, if *s* contains the sequence node, an additional cost of $O(s^2)$ is incurred. In summary, *op* is $O(d_{new} * d_{old})$ if a leaf node is updated, and $O(c_{max}' * c_{max} * h_s)$ if an internal node is updated. For updates involving the sequence node, *op* is $O(s^2 + c_{max}' * c_{max} * h_s)$ if an internal node is updated. In the special case when the updated internal node represents a record, *op* is $O(s^2 + c_{max}' * c_{max} * h_s + l_n+l_l+l_g+l_s+l_o)$.

Combining the *nav*, *op*, and *prop* times calculated above, the final time complexity for each maintenance operation is obtained. These results are shown in Table 3-5. Note that the

[4] This algorithm uses specifics of the XML schema used, knowledge about the "key" elements for identification purposes (in the presence of parent elements with same name), and an approximation to the edit distance.

Table 3-5. Time complexity of the quality-aware operations.

Operation	Type of node	Time for operation	Time for navigation	Time for propagation	Final complexity
Insert	Leaf node (non-seq)	$O(1)$	$O(p_l)$	$O(p_l * c_{max})$	$O(p_l * c_{max})$
	Leaf node (seq)	$O(s)$	$O(p_l)$	$O(p_l * c_{max})$	$O(s + p_l * c_{max})$
	Subtree (non-seq)	$O(n_s)$	$O(p_l)$	$O(p_l * c_{max})$	$O(n_s + p_l * c_{max})$
	Subtree (seq)	$O(n_s+s)$	$O(p_l)$	$O(p_l * c_{max})$	$O(n_s+s+p_l * c_{max})$
	Subtree (record)	$O(n+s+l_n+ l_t+l_g+l_s+l_o)$	$O(p_l)$	$O(1)$	$O(n+s+l_n+ l_t+l_g+l_s+l_o)$
Delete	Any node	$O(1)$	$O(p_l)$	$O(p_l * c_{max})$	$O(p_l * c_{max})$
Update	Leaf node	$O(d_{new} * d_{old})$	$O(p_l)$	$O(p_l * c_{max})$	$O(d_{new} * d_{old} + p_l * a_c)$
	Subtree (non-seq)	$O(c_{max}' * c_{max} * h_s)$	$O(p_l)$	$O(p_l * c_{max})$	$O(c_{max}' * c_{max} * h_s + p_l * c_{max})$
	Subtree (seq)	$O(s^2 + c_{max}' * c_{max} * h_s)$	$O(p_l)$	$O(p_l * c_{max})$	$O(s^2 + p_l * c_{max} + c_{max}' * c_{max} * h_s)$
	Subtree (record)	$O(s^2 + c_{max}' * c_{max} * h_s + l_n+l_t+l_g+l_s+l_o)$	$O(1)$	$O(1)$	$O(s^2 + c_{max}' * c_{max} * h_s + l_n+l_t+l_g+l_s+l_o)$

length p_l of a path p is always upper-bounded by the height of the tree. Assuming that the tree is roughly balanced, we have $p_l \leq \log(n)$.

CHAPTER 4
QUALITY MANAGEMENT ARCHITECTURE

The Quality Metadata Architecture (QMA) enables the integration of our Quality Estimation Model (QEM) with an existing biological repository. The key constraint in the design of the QMA was to minimize the changes that need to be made to the existing biological repository.

4.1 Usage Scenario

The usage scenario described here serves as a basis for the design and development of our quality management architecture, and illustrates what we predict to be the common integration approach of our quality estimation model with existing biological databases. The two main components we address are: 1) the source of biological data, and 2) the software layer that embeds our Quality Metadata Engine (or *QM-engine*, for short).

We assume that the biological information of interest resides in an external data source managed by an independent entity. Access to its contents is limited to what the source offers through its API. In addition, the owner of the data source may impose restrictions on when to query the database and how frequently. This external data source has its own user and administrator interface, through which data is queried and modified on a regular basis.

On the other hand, the QM-engine is managed separately by the quality providers and is considered to be 'local' to them (i.e., it runs on their servers). The QM-engine computes, stores, and updates the quality assessments of the underlying biological data. Such quality assessments or metadata are stored locally in a metadata source. Likewise, a local cache of the external data source is maintained by the QM-engine. A cache coherency strategy needs to be implemented such that changes to the original data in the external data source are replicated in the local cache. For this purpose, periodic update requests are sent to the external data source by the QM-engine.

Query processing is also part of the functionality of the QM-engine; it would enable the creation of a quality-augmented interface that allows users to quickly and easily retrieve quality metadata along with the biological data of interest. The User Interface is separate from the QM-engine, but it should generally be able to support quality specifications given by a user (e.g., rank results based on a particular quality dimension, show only a subset of the quality dimensions, and filter out results whose quality score is lower than certain threshold).

Following good software engineering practice, we modularize the QM-engine such that small modules handle specific tasks and interact with other modules through well-defined service interfaces.

4.2 Overview of Reference Architecture

The reference Quality Management Architecture is shown in Figure 4-1. The two main components of the usage scenario described in the previous section are represented by the External Data Source and the Quality Metadata Engine (QM-engine), respectively. The QM-engine contains a repository for the biological data that is locally cached (Local Cache) and a repository for the quality metadata that is locally-maintained (Metadata Source). At the core of the QM-engine is the Quality Layer, which handles data and metadata initialization, loading, refreshing, and maintenance. The Quality Layer is divided into five main modules: Data Manager, Metadata Manager, Maintenance Manager, XML Wrapper, and Query Processor. Next, we describe each of the conceptual components of the QMA and provide the functional specification for the modules in the Quality Layer.

4.2.1 External Data Source

The External Data Source is a repository of biological information with its own native API. Typically, there exists an Information Retrieval System with its own User Interface for providing queriable access to the source data.

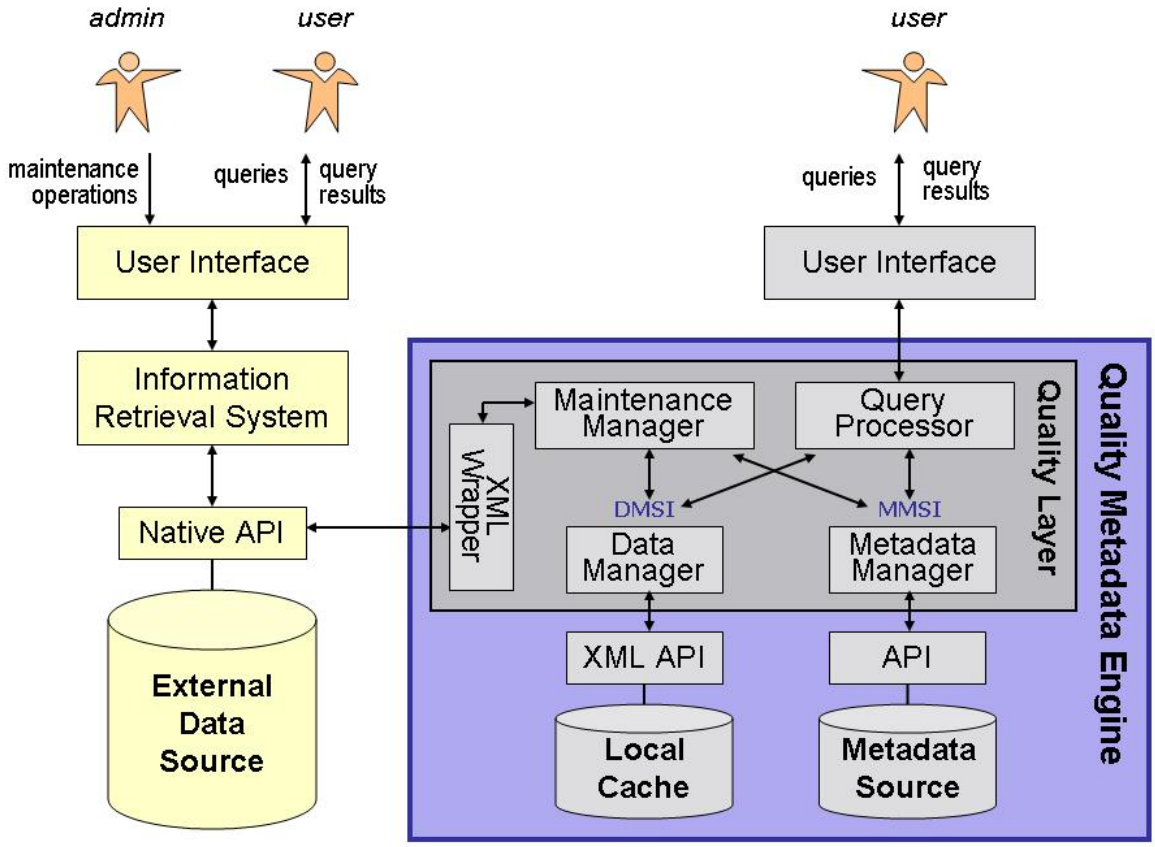


Figure 4-1. Reference Quality Metadata Architecture.

4.2.2 Quality Metadata Engine

4.2.2.1 Local cache

The Local Cache stores a subset of the data from the External Data Source in a local database. Data in the Local Cache is represented using the XML model, regardless of what the original representation of the biological data is at the External Data Source. Although we wanted to keep the QEM as general as possible, we adopted XML as its model since most of the popular genomics repositories provide support for XML. The conversion between the original data format and the XML format used in the Local Cache is handled by the XML Wrapper. The Local

Cache provides an XML API to the Quality Layer directly above. We have more to say about the specific XML implementation in Section 4.3.

4.2.2.2 Metadata source

The Metadata Source is a database that stores the quality metadata of the biological data in the Local Cache. It can either be a relational or XML-enabled database since both representations can accommodate well the quality metadata. The Metadata Source provides an API to the Quality Layer directly above.

4.2.2.3 Quality layer

The function of the Quality Layer is to manage a set of interacting software components which together maintain the quality metadata associated with the locally cached data, and provide access to both data and metadata. The Quality Layer therefore interacts with the API of the Local Cache and with the API of the Metadata Source.

The external inputs of this component are:

- A user request filtered by the User Interface.
- Data fetched from the External Data Source.

The external output of this component is a response to the user request to be displayed by the User Interface.

Data Manager. The function of the Data Manager is to control access to the Local Cache. Data can be queried or changed only through operations specified in the Data Manager Service Interface (DMSI).

The Data Manager interacts with the following components:

- With the Local Cache through the Cache's XML API.
- With the Query Processor and Maintenance Manager via the DMSI.

The tasks to be performed by the Data Manager are:

- On Query Processor's request, retrieve quality metadata from the Local Cache.
- On Maintenance Manager's request, update data in Local Cache.
- On Maintenance Manager's request, add data to Local Cache. If the Cache is full, evict a record, notify the Metadata Manager of the deletion, and insert the new data in the Cache.
- On Maintenance Manager's request, delete data from Local Cache.

Metadata Manager. The function of the Metadata Manager is to control access to the Metadata Source. Quality metadata can be queried or changed only through operations specified in the Metadata Manager Service Interface (MMSI).

The Metadata Manager interacts with the following components:

- With the Metadata Source through the Source's XML API.
- With the Maintenance Manager and Query Processor via the MMSI.

The tasks to be performed by the Metadata Manager are:

- On Query Processor's request, retrieve quality metadata from the Metadata Source.
- On Maintenance Manager's request, update quality metadata in the Metadata Source.
- On Maintenance Manager's request, add quality metadata to the Metadata Source.
- On Maintenance Manager's request, delete quality metadata from the Metadata Source.
- On Maintenance Manager's request, age quality metadata in the Metadata Source.

Maintenance Manager. The function of the Maintenance Manager is to maintain coherency between the Local Cache and the External Data Source, and to enforce consistency between contents of Local Cache and Metadata Source. It also handles cache misses.

The Maintenance Manager interacts with the following components:

- With the XML Wrapper.
- With the Data Manager through the DMSI.
- With the Metadata Manager through the MMSI.

The tasks to be performed by the Maintenance Manager are:

- Synchronize the Local Cache with the External Data Source on a regular basis with the help of the XML Wrapper.
- Handle Local Cache misses.
- Notify the Metadata Manager of any data updates (or deletions) to the Local Cache.

XML Wrapper. The function of the XML Wrapper is to fetch data from the External Data Source as needed. The external input of this component is the data fetched from the External Data Source. The internal interaction of this component is with the Data Manager. The task to be performed by this component is to fetch, upon Maintenance Manager's request, data from the External Data Source into a local working space and convert it to XML format.

Query Processor. The function of the Query Processor is to determine what data and metadata is needed to answer the user query, to request this information from the Data Manager and Metadata Manager, to combine the retrieved data and metadata to answer the user query, and to send back this response to the User Interface for display. The external input of this component is a user request filtered by the User Interface. The external output of this component is a response to the user request to be displayed by the User Interface.

The Query Processor interacts with the following components:

- With the User Interface.
- With the Data Manager through the DMSI.
- With the Metadata Manager through the MMSI.

The tasks to be performed by the Query Processor are:

- Serve query requests from the User Interface.
- Send data and metadata requests to the Data Manager and Metadata Manager, respectively, in order to execute a query.
- Combine the retrieved data and metadata in a meaningful way so as to answer the user request.

4.3 Architecture Implementation

4.3.1 Data Model: XML and Schema

In the previous section it was mentioned that data would be stored as XML in the Local Cache. Here, we provide the details of the specific XML format used in our implementation of the architecture.

4.3.1.1 Base XML format

The format we use to store data in the Local Cache is a modified version of the *INSDSeq* XML format. INSDSeq is the official supported XML format of the International Nucleotide Sequence Database Collaboration (INSD) [20], formed by DDBJ, EMBL, and GenBank. Figure 4-2 shows a fragment of a genomics record represented in this format. The INSD specifies the INSDSeq format using a *Document Type Definition* (DTD)[5]. However, an equivalent *XML schema*[6] can be obtained by means of the NCBI's DATATOOL [34], which is a utility program designed to convert *ASN.1*[7] specifications into XML DTD or XML schema, and DTD into XML schema. The latest version of the INSDSeq's schema (Version 1.4, 19 September 2005) automatically generated using DATATOOL (version 1.8.1, 18 January 2007) is also available online from the NCBI website [38]. The complete XML schema of the INSDSeq data format that we used as base for our XML format is presented in Appendix C.

4.3.1.2 Modified XML format

The original INSDSeq format was modified to fit the needs of our Quality Management Architecture implementation. In particular, the following four amendments were made: 1) the addition of an *id* attribute to every element node, 2) the exclusion of the *INSDSet* element, 3) the inclusion of schema annotations into the XML schema, and 4) the substitution of the DTD declaration for the schema declaration in every record document. We briefly describe the motivation behind each of these modifications next. Figure 4-3 shows an example record

[5] A DTD provides a grammar for a class of documents [62].

[6] XML Schema provides a means for defining the structure, content and semantics of XML documents (W3C Recommendation 28 October 2004) [63].

[7] ASN.1 or *Abstract Syntax Notation number One*, is an International Standards Organization (ISO) standard for describing structured data; it is a formal language for the specification of abstract data types [18].

```
<?xml version='1.0' ?>
<!DOCTYPE INSDSet PUBLIC "-//NCBI//INSD INSDSeq/EN"
"http://www.ncbi.nlm.nih.gov/dtd/INSD_INSDSeq.dtd">
<INSDSet>
<INSDSeq>
  <INSDSeq_locus>AF030859</INSDSeq_locus>
  <INSDSeq_length>4065</INSDSeq_length>
  <INSDSeq_strandedness>double</INSDSeq_strandedness>
  <INSDSeq_moltype>DNA</INSDSeq_moltype>
  <INSDSeq_topology>linear</INSDSeq_topology>
  <INSDSeq_division>PLN</INSDSeq_division>
  <INSDSeq_update-date>21-SEP-1998</INSDSeq_update-date>
  <INSDSeq_create-date>22-SEP-1998</INSDSeq_create-date>
...
<INSDSeq_feature-table>
    <INSDFeature>
      <INSDFeature_key>source</INSDFeature_key>
      <INSDFeature_location>1..4065</INSDFeature_location>
      <INSDFeature_intervals>
        <INSDInterval>
          <INSDInterval_from>1</INSDInterval_from>
          <INSDInterval_to>4065</INSDInterval_to>
          <INSDInterval_accession>AF030859.1</INSDInterval_accession>
        </INSDInterval>
      </INSDFeature_intervals>
...
</INSDSeq_feature-table>
...
</INSDSeq>
</INSDSet>
```

Figure 4-2. Example of a record represented in the INSDSeq XML format (only a fragment is shown).

```
<?xml version='1.0' ?>
<INSDSeq xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xmlns="http://biodq.cise.ufl.edu"
     xsi:schemaLocation="http://biodq.cise.ufl.edu
     http://biodq/schemas/INSDSeq_QM.xsd"
     id="i1" >
  <INSDSeq_locus id="i2">AF030859</INSDSeq_locus>
  <INSDSeq_length id="i3">4065</INSDSeq_length>
  <INSDSeq_strandedness id="i4">double</INSDSeq_strandedness>
  <INSDSeq_moltype id="i5">DNA</INSDSeq_moltype>
  <INSDSeq_topology id="i6">linear</INSDSeq_topology>
  <INSDSeq_division id="i7">PLN</INSDSeq_division>
  <INSDSeq_update-date id="i8">21-SEP-1998</INSDSeq_update-date>
  <INSDSeq_create-date id="i9">22-SEP-1998</INSDSeq_create-date>
...
<INSDSeq_feature-table id="i43">
   <INSDFeature id="i44">
     <INSDFeature_key id="i45">source</INSDFeature_key>
     <INSDFeature_location id="i46">1..4065</INSDFeature_location>
     <INSDFeature_intervals id="i47">
       <INSDInterval id="i48">
         <INSDInterval_from id="i49">1</INSDInterval_from>
         <INSDInterval_to id="i50">4065</INSDInterval_to>
         <INSDInterval_accession id="i51">AF030859.1</INSDInterval_accession>
       </INSDInterval>
     </INSDFeature_intervals>
...
</INSDSeq_feature-table>
...
</INSDSeq>
```

Figure 4-3. Example of a record represented in the INSDSeq_QM XML format (only a fragment is shown).

represented in the modified INSDSeq format, and Appendix D contains the modified XML schema.

The first modification was needed in order to associate the quality metadata stored in the Metadata Source to the data stored in the Local Cache. Specifically, an identifier was required for every element node in the data tree (XML document) of every record in the Local Cache, so that the same identifier could be attached to the corresponding quality metadata in the Metadata Source. Since records from the NCBI databases already possess an identifier called *accession number*[8], we used a combination of the accession number and a consecutive number assigned to every node, as our element identifier. For this purpose, we added 'id' values that would contain the consecutive number associated to each element node in the document. The type of these identifiers is *xs:ID*, as defined by the W3C Recommendation of September 9, 2005 [64]. Inclusion of the 'id' attributes affects both the schema and *instance*[9] documents (see Figure 4-3 and Appendix D).

The second modification, which consisted of the exclusion of the *INSDSet* element, was not strictly required but yielded a cleaner data model. In the original format (see Appendix C), the INSDSet element simply acts as a container for *INSDSeq* elements, which are the really relevant elements because they represent a record. When an XML document contains a single record, the presence of the INSDSet element obscures the fact that there is only one record since one would expect to see more than one component element (of type INSDSet) in the set element (of type INSDSet). In that case, it seems more natural to remove the INSDSet element and leave the INSDSeq element as the principal element (i.e., the document's root element). We consider

[8] The accession number is the record identifier used across the NCBI databases.

[9] Instance documents are XML documents that conform to a particular schema, i.e., can be validated against it.

that the INSDSet element is useful for grouping records but it is not so convenient in our context, where the manipulation (i.e., insertion, deletion) of records will be done on an individual basis. Given that no group or set of records will be inserted as an entity per se in the Local Cache, we decided to remove the INSDSet element from our schema. Deletion of this element affected both the schema and instance documents (see Figures 4-2, 4-3 and Appendix D).

The third modification, namely the insertion of schema annotations into the XML schema, was necessary for customizing the generation of objects, types, and tables during the schema registration process in *Oracle XML DB*[10] (Oracle XML DB was chosen as the DBMS for the Local Cache) [2]. The schema annotations also control how instance XML documents get mapped to the database. Details of the annotations will not be discussed here, but can be found in Appendix D. Inclusion of these annotations has a direct effect on the schema document only.

The fourth and last modification, namely the substitution of the DTD declaration for the schema declaration in every record document, was made in order to use the structured storage of XML documents in Oracle XML DB. Using the structured storage option, an XML document is shredded and stored as a set of SQL objects (the exact mapping depends on the options specified when the XML schema is registered) rather than as a *Character Large Object* (CLOB), and this has several advantages including optimized memory management, reduced storage requirements, B-tree indexing, and in-place updates [2,16]. Since structured storage is only available when the *XMLType*[11] table has been constrained to an XML schema, the XML documents we insert must necessarily be instance documents of the schema associated to the Oracle table, and as such, should declare the schema they conform to. Hence, there is no room for the DTD in this context,

[10] Oracle XML DB is a set of Oracle Database technologies related to high-performance XML storage and retrieval [2, 16].

[11] XMLType is a datatype used in Oracle to define columns or tables that contain XML.

and we have to replace the DTD definition line in every record document for the appropriate schema declaration. This modification affected instance documents only. See Figures 4-2 and 4-3 for an illustration of how a record in the original INSDSeq format refers to a DTD while in the modified format refers to a schema.

4.3.2 External Data Source: the NCBI's Nucleotide and Protein Databases

The External Data Source was previously described as a repository of biological information with its own programmatic and user interface. In the context of our QMA implementation, we selected the NCBI's Nucleotide and Protein databases as our External Data Source. The widespread use of these databases by the scientific community, the availability of versioning information (in the form of revision history) about the stored records, and the availability of relatively friendly programmatic interfaces, were reasons that influenced our decision.

The Nucleotide database contains all the sequence data from GenBank, EMBL, and DDBJ, the members of the International Nucleotide Sequence Database Collaboration. The Nucleotide database is divided into three main subsets: EST (Expressed Sequence Tags), GSS (Genome Survey Sequences), and CoreNucleotide (which comprises the remaining nucleotide sequences not in EST or GSS). On the other hand, the Protein database contains sequence data from translated coding regions from DNA sequences in GenBank, EMBL, and DDBJ as well as protein sequences submitted to Protein Information Resource, SWISS-PROT, Protein Research Foundation, and Protein Data Bank [41].

4.3.2.1 Entrez Retrieval System and Interface

Records from the major NCBI databases (including the nucleotide and protein databases) are accessible via Entrez [61, 36, 40], the NCBI retrieval system. The Entrez system covers about 91 million nucleotide and protein sequence records from several sources (as of 2006) [61].

Records can be retrieved from Entrez in different formats (e.g., XML, ASN.1, FASTA, flat file) and downloaded singly or in batches. There is a Web-based interface to the Entrez system named Global Query, which is the default the default search engine on the NCBI homepage [61]. There is also a programmatic interface to Entrez, provided by the Entrez Programming Utilities (a.k.a. E-Utilities or eUtils) [61, 40], which is a suite of eight server-side programs used to search, link between, and download from, the Entrez databases.

4.3.2.2 FTP Interface

Many of the resources (data and tools) that NCBI provides are available by FTP. Complete bimonthly releases and daily updates of the GenBank (nucleotide) and RefSeq (nucleotide and protein) databases are available from the NCBI FTP site (ftp://ftp.ncbi.nih.gov/) [5]. GenBank releases are distributed in compressed flat-file and ASN.1 formats. RefSeq releases are distributed in compressed flat-file format as well as in binary format.

4.3.3 Local Cache: an XML Database

Previously, the Local Cache was described as a database capable of handling semistructured data. Here we provide details about how the Local Cache is actually implemented. We chose *Oracle XML DB* (10*g* Release 2) [47] as the DBMS for the Local Cache after examining freely-available databases that provided XML support, including eXist [27], Montag [8], MonetDB [13]. Oracle XML DB was selected because it offered many options for storage, APIs, content management, as well full support for *XQuery*[12], *SQL/XML*[13], and SQL functions for updating XML [2]. As opposed to newer XML databases, Oracle XML DB represented a relatively mature product with stable releases, comprehensive documentation, and the support of a world's leading database company.

[12] XQuery is an XML query language (W3C Recommendation 23 January 2007) [65].

[13] SQL/XML is an SQL standard for XML (ISO/IEC 9075-14:2005 (E) draft, May 2005).

4.3.3.1 XML schema registration

Since the data managed by the Local Cache has an associated XML schema (see Appendix D), we use the structured storage option available in Oracle XML DB, which persists XML data (i.e., instance documents associated to a schema) into a set of relational tables created when the schema is registered with Oracle. Our *INSDSeq_QM* schema was registered with Oracle under the XML schema URL[14] *http://biodq/schemas/INSDSeq_QM.xsd*, which has to be referenced by every instance document that we insert in the database. Such reference is contained in the schema declaration line of the instance documents, specifically in the *xsi:schemaLocation* attribute (see Figure 4-3) which is used when the referenced schema declares a *targetNamespace* (in our case, the targetNamespace of the schema is *http://biodq.cise.ufl.edu*).

4.3.3.2 Table creation

After registering our XML schema with the database, we created an *XMLType* table named *CACHE*, constrained to the global element *INSDSeq* defined by the *INSDSeq_QM* registered schema. In Oracle XML DB, when a table or column of type *XMLType* has been constrained to a particular element and a particular XML schema, it can only contain documents that are compliant with the schema definition of that element. In our case, the *CACHE* table can only store instance documents of the *INSDSeq_QM* schema.

4.3.3.3 Cache replacement strategy

Per definition, the Local Cache is of limited size, which in most cases does not permit the storage of the entire source database. The cache size is a parameter that can be adjusted according to the expected system usage. We have to address the issue of what cache replacement strategy to use whenever the Local Cache becomes full and new data needs to be added. In our

[14] The XML schema URL is a unique identifier for an XML schema, internally used by Oracle XML DB.

current implementation, the *Clock Algorithm* (also known as *Second Chance*) is used because it is an efficient approximation to the *Least Recently Used* (LRU) strategy [19].

The Clock Algorithm works as follows. The records (buffers or pages, in other contexts) are arranged in a circle (in practice, a circular list), and each has an associated "flag" (or bit) which is either 0 or 1. When a record is added to the cache, its flag is set to 1. Similarly, when a record is accessed, its flag is set to 1. There is a "hand" that always points to one of the records, and rotates clockwise when searching for a victim record (to throw out of the cache). If a victim record is needed (i.e., if the cache is full and a new record needs to be inserted), the hand looks for the first record with a 0 flag, rotating clockwise. When it sees flags equal to 1, it sets them to 0. In this way, a record is only thrown out of the cache if it remains unaccessed for the time it takes the hand to make a complete rotation to set the record's flag to 0 and then make another complete rotation to find the record's 0 flag unchanged.

Alternative cache replacement strategies include *First In, First Out* (FIFO), *Most Recently Used* (MRU), LRU and random. Each of them has advantages and disadvantages with respect to efficiency (algorithmic complexity), book-keeping overhead, and thrashing behavior. The Clock Algorithm chosen for the QMA has the advantage of being efficient, simple, and having low overhead (only one 'bit' per record is needed), but its worst case (or thrashing) behavior occurs for sequential scans of the data. In Chapter 6 we mention that a direction for future research is the development of cache replacement strategies optimized for the typical access patterns of biological data.

4.3.4 Metadata Source: a Relational Database

The Metadata Source was previously described as a database for storing the quality metadata of the biological data. Here, we give out details regarding the type of database and logical representation of the quality metadata within this database. In our implementation, *Oracle*

Database 10*g* [14] was chosen as the DBMS for the Metadata Source. We selected it for consistency across DBMS of the Local Cache and Metadata Source; it was not convenient for us to have two different DBMS within the QM-engine, especially if we could accomplish the same task with only one. Although both the Metadata Source and the Local Cache are managed by the Oracle DBMS, a key difference between these databases is that the former uses a relational schema for the data whereas the latter uses an XML schema. Hence the Metadata Source is implemented as a relational database in Oracle.

4.3.4.1 Relational schema

The relational schema used for the quality metadata (shown in Figure 4-4) consists of two tables named *QM_NODES* and *QM_DOC*. The QM_NODES table contains the quality metadata of XML nodes stored in the Local Cache, so that a row in this table corresponds to a node from a cached instance XML document. On the other hand, the QM_DOC table contains the quality metadata associated to instance documents; in particular, in contains the quality metadata of the document's root node, which summarizes the metadata of all other nodes in the tree. It also contains metadata that is applicable to the document only (not to the nodes that compose it). Next, we describe in more detail what each of these tables contains.

The QM_NODES table. A row in this table represents the quality metadata of a node in an instance document stored in the Local Cache. This table contains the following eight fields or columns:

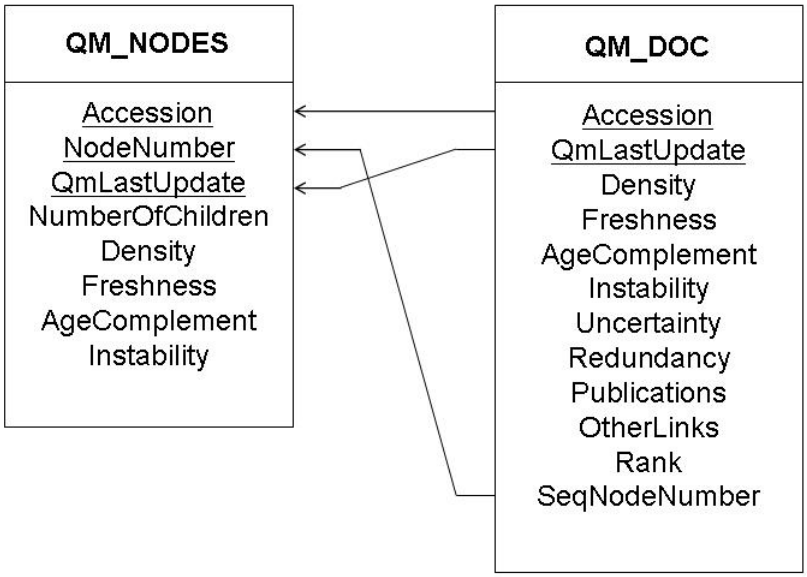


Figure 4-4. Relational schema for the quality metadata in the Metadata Source.

- *Accession*: accession number identifying the instance document to which the node belongs.
- *NodeNumber*: number that identifies the node within the instance document; it is the value of the node's *id* attribute.
- *QmLastUpdate*: date in which the quality metadata of the node was last updated.
- *NumberOfChildren*: number of child nodes that the node has.
- *Density*: score for the Density dimension of the node (defined in Chapter 3).
- *Freshness*: score for the Freshness dimension of the node (defined in Chapter 3).
- *AgeComplement*: complement of the age score of the node, i.e., one minus the age score.
- *Instability*: complement of the stability score of the node, i.e., one minus the stability score.

The primary key of the QM_NODES table is composed of three fields: *Accession*, *NodeNumber*, and *QmLastUpdate*. A node in an instance document can be identified using *Accession* and *NodeNumber* alone, but *QmLastUpdate* is needed to differentiate among quality metadata entries (for the same node) computed on different dates, which facilitates update operations and recovery at the application level.

The QM_DOC table. A row in this table represents the quality metadata of an instance document stored in the Local Cache. This table contains the following twelve fields or columns:

- *Accession*: accession number identifying the instance document.
- *QmLastUpdate*: date in which the quality metadata of the document was last updated.
- *Density*: score for the Density dimension of the document's root node.
- *Freshness*: score for the Freshness dimension of the document's root node.
- *AgeComplement*: complement of the age score of the document's root node.
- *Instability*: complement of the stability score of the document's root node.
- *Uncertainty*: score for the Uncertainty dimension of the document.
- *Redundancy*: score for the Redundancy dimension of the document.
- *Publications*: score for the Publications dimension of the document.
- *OtherLinks*: score for the OtherLinks dimension of the document.
- *Rank*: overall ranking or score of the document.
- *SeqNodeNumber*: identifier of the sequence node within the document; value of the sequence node's *id* attribute.

The primary key of the QM_DOC table is composed of two fields: *Accession* and *QmLastUpdate*. An instance document can be identified using *Accession* alone, but *QmLastUpdate* is needed, as in QM_NODES, to differentiate among quality metadata entries computed on different dates (for the same document), which facilitates update operations and

recovery at the application level. The QM_DOC and QM_NODES tables are related through a foreign key that consists of the fields *Accession*, *QmLastUpdate*, and *SeqNodeNumber*. Recall that in our context the sequence node is a leaf node, and it contains the nucleotide or protein sequence of the biological record. The reason why we have a pointer (foreign key) to the quality metadata of this particular node from the document's quality metadata is because domain experts regard the sequence data as a highly valuable piece of information, and they pay much attention to changes applied to the sequence. Therefore, based on experts input, we decided that it was convenient to have a quick way to retrieve the quality metadata associated to this special node from the document's metadata (since users may query this information often).

4.3.4.2 Index creation

We created two indexes, *QM_NODES_accession_index* and *QM_DOC_accession_index*, over the *Accession* column of the QM_NODES and QM_DOC tables, respectively. The aim of these indexes was to speed up accesses to quality metadata associated to a biological record identified by an accession number. When an update is executed or a user query is processed, the QM-engine typically needs to retrieve the quality metadata of nodes belonging to an instance document (record) identified by a particular accession number, so having an index over the accession number in each table can speed up these frequent operations.

4.3.5 Quality Layer: Java Classes

We previously mention that the Quality Layer manages a set of interacting software components that together maintain the quality metadata of the data cached locally. Here we describe how this layer and its components are implemented.

The Quality Layer is implemented by a set of Java classes which closely match the components outlined in Figure 4-1. The JRE version used to build and run our project was version 1.6.0-b105. The classes implementing the Quality Layer are

- *QualityLayerService*: this class encloses various component classes that interact with the purpose of providing quality information about the data cached from the External Data Source. This class provides methods for starting the service, stopping the service, as well as for handling a user's request.

- *DbObject*: this class encapsulates a real database and offers a standard set of JDBC (Java Database Connectivity) operations to interact with the database, as well as a special method for handling XML data. It provides methods for connecting to the database, executing queries, executing updates (inserts, updates, or deletes), closing and resetting the database connection, setting the commit mode, as well as commit and rollback methods.

- *DataManager*: this class controls access to the database object representing the Local Cache. This database object is set to auto-commit by default. The DataManager class provides methods for inserting, deleting, and retrieving XML instance documents to/from the database. It also has start and shutdown methods.

- *MetadataManager*: this class controls access to the database object representing the Metadata Source. This database object has the auto-commit property turned off by default since we want to be able manage insert and update operations as atomic operations, i.e., either we insert/update the quality metadata of all nodes within an instance document, or we do not carry out the operation. So we need to handle transactions and be able to commit or rollback operation at the application level. The MetadataManager class provides methods for inserting, deleting, and retrieving quality metadata to/from the database. It also has start and shutdown methods. Finally, it provides commit and rollback methods.

- *MaintenanceManager*: this class is in charge of updating the Local Cache and Metadata Source upon changes in the External Data Source. It provides methods for performing daily maintenance (including update and aging of records), and initial loading of records (which needs to retrieve all previous versions of records). It also has start and shutdown methods.

- *QueryProcessor*: this class processes and executes queries by calling methods from the Data Manager and Metadata Manager to retrieve the necessary information. It provides methods for processing queries, start and shutdown.

- *XMLWrapper*: this class acts as the interface between the External Data Source and the Quality Layer by fetching data and converting it to the appropriate format for internal use within the Quality Layer. It provides methods for retrieving data (with and without version history) and links from the NCBI database, as well as checking for daily updates.

- *EUtils*: this class handles calls to the External Data Source's interface, E-Utilities, and passes the results from these calls to the *XMLWrapper*. It provides methods for calling the eSearch, eFetch, eLink, eGQuery, and eSummary programs.

- *QualityMetadata*: this class represents the quality information that is computed and stored about a node from an XML instance document. It has accessor methods (i.e., get and set methods) for all the quality dimensions included in the quality metadata of a node.

- *QualityMetadataDoc*: this class represents the quality information that is computed and stored about an XML instance document. It has accessor methods (i.e., get and set methods) for all the quality dimensions included in the quality metadata of a document.

- *BatchXMLProcessor*: this class parses the XML documents that contain batches of records from the External Data Source, and produces a single xml document per record. These smaller documents are slightly modified so that they become instance documents of our *INSDSeq_QM* XML schema, and then passed onto the *MaintenanceManager* for further processing. It uses SAX (Simple API for XML), which is an event-driven push model for processing XML, and a de facto standard.

- *LinkXMLProcessor*: this class parses the XML link data output by the NCBI's ELink utility program, and extracts counts of publication links, neighbors, and other-links. It uses SAX.

- *Utils*: this is a utility class where public constants and generic methods are defined so that they become available to other classes.

4.3.6 Macro Level Operations Implemented in the QMA

4.3.6.1 Bulk-loading

Bulk-loading takes an input data set from the external data source (in our case, the NCBI databases) and loads it into the QMA system. This process would typically be run during the early stage of the integration with the external data source, but it can also be used to load additional data sets after the initial integration. Bulk-loading involves the computation of all quality scores, the storage of this quality metadata in the local metadata source, and the storage of the biological data in the local data cache.

The steps involved in bulk-loading a subset of the NCBI repository into the prototype QMA system are described next. First, a file containing the accession numbers of the records to bulk load is read by the Maintenance Manager component of the QMA. Then, the Maintenance Manager asks the XML Wrapper to retrieve these records (and their previous versions, if available) from the NCBI repository. The Wrapper sends batch queries to the NCBI database server, via the NCBI's E-Utils programming interface, until all records are retrieved. Once the data has been downloaded, the Maintenance Manager starts processing it, reading one batch-file

at a time. If various versions of the same record are available, the records are processed in ascending order of version number (i.e., oldest version first).

The processing of a record varies depending on whether it is the first version of a record or not. If the record to process is the first version of a record, the Initialize function is used; otherwise the Update function is used. The Initialize function parses a record and computes its initial quality scores. Then, the record is passed to the Data Manager for insertion into the Local Cache, and its quality scores are passed to the Metadata Manager for insertion into the Metadata Source. On the other hand, the Update function first has to locate the previous version (and associated metadata) of the record to update, and to do this it sends requests to the Data Manager and Metadata Manager, which search for the pertinent data in the Local Cache and Metadata Source, respectively. After both data and metadata from the previous version of the record are found, the Update function parses the older and newer versions of the record, comparing their contents, and updating the quality scores as needed. Then, the updated record is passed to the Data Manager for insertion into the Local Cache, and its updated quality scores are passed to the Metadata Manager for insertion into the Metadata Source. The Data Manager deletes the older version of the record and inserts the new one such that only the most recent version of the record is kept in the cache. Likewise, the Metadata Manager deletes the older metadata of the record and inserts the updated metadata that corresponds to the newer version of the record. This processing of records continues until all downloaded records have been loaded into the QMA.

Finally, the records are "aged" so that the quality scores reflect the time elapsed since the records were last updated. The Aging function (not to be confused with the quality dimension named Age) updates the scores of the temporal dimensions only i.e., Freshness, Stability, and

Age. This aging process is also referred to as metadata *refreshing*, to differentiate it from the metadata *update*.

4.3.6.2 Maintenance

Maintenance refers to the process of keeping the contents of the QMA updated with respect to changes in the external data source (the NCBI repository, in our case). Ideally, the data and metadata stored in the QMA should be updated as frequently as the external data source is. In our prototype QMA system, maintenance is performed on a daily basis since the NCBI databases we use (GenBank and RefSeq) provide incremental daily updates, available from the NCBI FTP site. Maintenance involves the detection of data changes in the external data source, the update of data in the local cache to enforce consistency with respect to the external database, and the update of the corresponding quality scores in the metadata source.

The steps involved in performing the daily maintenance of the prototype QMA system are described next. First, the Maintenance Manager asks the XML Wrapper to check if updates are available. The Wrapper then connects to the NCBI's FTP site and checks if daily updates are available for download. If no updates are found, maintenance proceeds to the aging phase. If updates are available, the Wrapper downloads them, uncompress them, and converts them to the appropriate XML format (GenBank's incremental updates are in ASN.1 format, and RefSeq's incremental updates are in binary format). Next, the Maintenance Manager starts processing the downloaded updates.

Incremental updates are scanned for accession numbers of records currently stored in the QMA. If no match is found, i.e., if none of the records stored in the Local Cache was updated in the NCBI repository, then the Maintenance Manager proceeds to the aging step. On the contrary, if one or more matches are found, the Update function (described in the previous section) is called on each such record to handle the update of both data and quality scores.

Once all updates are processed, the aging phase follows. Records that were not updated are *aged*. Aging or *refreshing*, as previously described, consists in updating the quality scores of the temporal quality dimensions of a record to reflect the time elapsed since the record was last updated.

Although updates represent the core of the QMA maintenance process, the *aging* of metadata is also an important part of maintenance. Depending on the number of records stored in the QMA, daily maintenance could be expensive due to the refreshing of non-updated records (even if no records are updated in the external data source). However, we expect that the size of the Local Cache used in practice be small enough to avoid this concern. Furthermore, the cost of aging a record is much smaller than the cost of updating it because only the temporal dimensions are changed (hence avoiding expensive computations like the ones required for the Stability measure, for example).

4.3.6.3 Querying

Querying in the QMA refers to the processing of user requests to retrieve specific data and metadata. The steps involved in processing of queries in the prototype QMA system are described next

First, the Query Processor determines what metadata needs to be retrieved based on the query filter, an optional parameter that specifies which quality dimensions are requested by the user (if the filter is omitted, all quality dimensions are retrieved by default). The Query Processor then forwards appropriate requests to the Data and Metadata Managers in order to retrieve the requested data and metadata. Finally, the information returned by the Data and Metadata Managers is combined and send back to the user.

CHAPTER 5
EVALUATION

5.1 Objectives

The main purpose of this evaluation is to determine the biological significance of the Quality Estimation Model and to validate the usefulness of the Quality Management Architecture. In particular, we focus on the following aspects:

- The usefulness of the chosen quality dimensions in assessing the quality of genomic data.
- The ability of the quality estimates produced by the Quality Estimation Model to discriminate high versus low quality data.
- The relevancy of the capabilities offered by the Quality Management Architecture.
- The usefulness of the implemented QMA prototype system with respect to its operational cost.

5.2 Evaluation of the Quality Estimation Model

5.2.1 Challenges of the Evaluation

The evaluation of our Quality Estimation Model was challenging in the following ways. The first challenge we faced was the lack of a standard benchmark for testing our quality estimates of biological data. To the best of our knowledge, there are no public genomic data sets with comprehensive quality assessments comparable to ours, thus making the evaluation of our model difficult. Biologists are also mainly unaware of the existence of genomic databases with quality scores (see Table B-6 of Appendix B), as found in our research study that is described in Section 5.2.2.1. The only two exceptions are the Greengenes [15] and SGN [32] databases, which offer quality information but are limited to only a few organisms and to very specific sequences (16S rRNA genes in the case of Greengenes and markers in the case of SGN). To overcome this problem and be able to obtain quality assessments for a variety of genomic data,

we consulted several experts in the fields of biology and genomics about the quality of genomic records.

The second challenge was to understand the biologists' perception of data quality, and to bridge the gap between their perspective and our needs for the purpose of the model evaluation. In a preliminary attempt to evaluate our model, we recruited five volunteer Ph.D. students majoring in biology-related fields at our university, and asked them to provide scores along four quality dimensions for 20 genomic records (selected by a collaborator from our university such that both high and low quality records were represented in the sample). This attempt was unsuccessful because only three of the subjects responded, and they did not provide the responses we sought. In particular, only one subject provided scores for the quality dimensions, and several records were assigned the same score (e.g., 0, 0.25, 0.5, etc.), indicating the use of an underlying discrete scale or categorization. The other two subjects responded that it was difficult to assign fine-grained scores along the given dimensions, and that the given set of records were not part of their area of expertise.

The lessons learned some from this first attempt were: *i*) biologists tend to express their quality assessments in a *discrete* scale rather than in a *continuous* scale, *ii*) it is difficult for biologists to confidently evaluate the quality of unfamiliar records (i.e., not from their area of expertise). These lessons greatly influenced the design of a second research study that we performed (see Section 5.2.2.1), in particular the use of a binary scale for quality assessments, and the freedom to choose the records to be assessed (rather than using a fixed set of records for all the participants). The challenge still remaining was devising a way to compare the continuous quality scores produced by our model with the discrete quality assessments provided by the experts.

The third challenge was due to the practical limitation in the size of the data set for which quality assessments could be obtained from domain experts. A single expert can reasonably be asked to evaluate the quality of only a handful of records given the time it takes to do so. Evaluating the quality of genomic records is a time consuming task because of the number of aspects that are considered by experts when making such assessments (examples can be found in Table B-3 from Appendix B). Aspects commonly considered are: amount of annotations, number of publications, quality of referenced journals, consistency of annotations, completeness of the sequence, number of revisions, and specific information about the gene such as exon/intron regions, areas of expression, structure, and function. On the other hand, finding domain experts willing to provide quality assessments even for a small number of records proved to be challenging. Hence, we are restricted in the number of domain experts and in the number of records evaluated per expert. However, in order to obtain meaningful results, it was necessary to evaluate our model over a reasonably large data set for which quality assessments were needed.

5.2.2 Experiments and Results

A series of experiments were conducted, some of them derived from findings in previous experiments; therefore we present each experiment along with its data set, results, and discussion separately. The core of our experimental evaluation is a research study conducted among several domain experts, which we describe next.

5.2.2.1 Research study

We conducted a research study among sixteen field-experts from universities in the United States during the months of March to June, 2007. This study was approved by the University of Florida Institutional Review Board on March 26, 2007 as Protocol #2007-U-0304, titled "Comparative Study of Automated and Human-based Quality Assessments over Genomic Data."

Purpose of the study. The purpose of the research study was to collect experts' assessments about the quality of genomic records. This information was to be used to evaluate the validity and usefulness of the quality estimation model for genomics data.

Participants of the study. Sixteen participants were recruited for our research study. All of them met the following eligibility requirements: 1) were 20 years old or older, 2) were Ph.D. students, post-docs, professors, or researchers in a university, 3) had background in areas related to biology or genomics, and 4) had experience using NCBI databases. Participation in this study was completely voluntary, and each participant received $100 compensation after completing the study questionnaire.

Description of the study. The research study questionnaire consisted of two parts. In the first part, participants were asked to choose a total of 24 records from NCBI's nucleotide or protein databases, divided in two sets as follows. The first set had to contain 12 records whose quality the participant considered to be "good" (above average). The second set had to contain 12 records whose quality the participant considered to be "poor" (below average). The participant provided the *accession numbers* of the records they chose (not the actual record contents). In the second part, participants were asked to answer a few questions regarding the criteria they used for evaluating the quality of genomics records, the relative quality of different databases from NCBI, and the suitability of quality assessment factors in the context of genomics.

When was the study performed? The recruiting of participants started in April 2007 and continued until June 2007. We collected responses from the participants during the period of May to June 2007.

How was the study conducted? The questionnaire of the research study (also referred to as *survey questionnaire*) was made available online to facilitate its access by the participants and

to automate the collection of responses. Our survey questionnaire was hosted at the SurveyMonkey website (URL http://www.surveymonkey.com), and access to it was restricted by password. A PDF version of our web-based survey is shown in Appendix A, with each page showed as a separate image.

Responses collected from the study. The responses given by the participants of our study can be found in Appendix B. The responses to each question are shown in separate tables, and each question is reproduced (from Appendix A) in the table heading for convenience. For confidentiality purposes, the participants were assigned a number ranging from 1 to 16; and the answers were associated with this identifier rather than with the name of the participants.

5.2.2.2 Experiment 1

Our initial set of quality dimensions included Density (without its two sub-dimensions), Stability, Freshness, Linkage (without its four sub-dimensions), and Redundancy (defined slightly differently). The relevance of these five dimensions was preliminary verified by expert collaborators from our university, but a more systematic approach was sought. This experiment aimed to evaluate the relevancy of our set of quality dimensions for the biologists.

Methodology. To evaluate the biological relevance of our initial set of quality dimensions, we asked our research participants to evaluate the usefulness of those dimensions in assessing the quality of genomic data. This was done in Part III of the survey questionnaire (see Appendix A), where participants had to classify each dimension in one of the following three categories: very useful, somewhat useful, or not useful. They also had to briefly comment on their choice. A description and an illustrative example of each dimension were provided to the participants so that there was common understanding about the dimensions being evaluated.

Additionally, we wanted to know if there were other relevant dimensions that could be added to the initial set of quality dimensions. For this purpose, we asked our research

participants to describe the criteria they used when evaluating the quality of a genomic record. This was done in question 1 of Part II of the survey questionnaire (see Appendix A). To avoid influencing the participants with our choice of dimensions, this question was asked before introducing our dimensions to them.

Results and Discussion. The responses to the questions in Part III of the survey questionnaire are shown in Tables B-7, B-8, B-9, B-10, and B-11 from Appendix B. Summarized results are presented in Table 5-1 and Figure 5-1. Table 5-1 shows the percent of responses in each category (very useful, somewhat useful, and not useful) for the five quality dimensions. Figure 5-1 illustrates the usefulness ratings from Table 5-1 using a stacked bar chart.

In Figure 5-1 we observe that at least 50% of the participants considered the Linkage, Density, and Freshness dimensions to be very useful in assessing the quality of genomic data. Linkage was the best-rated dimension, while Redundancy was the worst-rated dimension. Another important observation is that 80% or more of the participants considered all but one of the dimensions (namely Redundancy) to be at least somewhat useful in assessing genomic data quality, which validates the relevancy of these dimensions (i.e., Linkage, Density, Freshness, and Stability). The usefulness of the Redundancy dimension is not strongly supported by the participants, since only 57% of them considered it to be at least somewhat useful. However, rather than dropping this dimension from the initial set, we revised its definition so that it would better reflect the biological notion of redundancy at the sequence level (rather than at the annotations level).

Now we discuss the results for the second part of the experiment. The responses to question 1 in Part II of the survey questionnaire are shown Table B-3. A summarized interpretation of the textual responses from Table B-3 is presented in Table 5-2 and Figure 5-2.

Table 5-1. Participants' assessment of the usefulness of the initial quality dimensions. The usefulness of the dimensions was evaluated in a 3-point scale: very useful (V), somewhat useful (S), and not useful (N). The percent of responses obtained in each category is shown.

Category	Linkage	Density	Freshness	Stability	Redundancy
V	81%	50%	50%	44%	13%
S	13%	38%	31%	38%	44%
N	6%	13%	19%	19%	44%

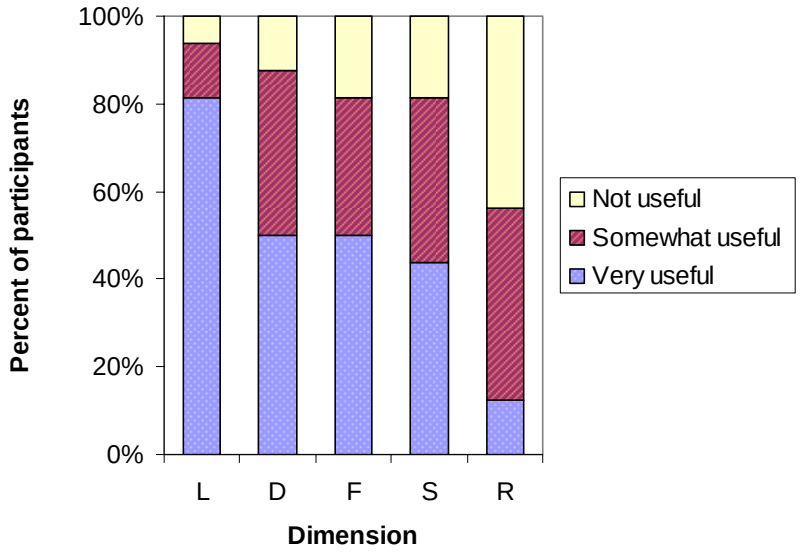


Figure 5-1. Assessment of the usefulness of the initial quality dimensions by the participants. The usefulness of the dimensions was evaluated in a 3-point scale: very useful, somewhat useful, or not useful. L stands for Linkage, D for Density, F for Freshness, S for Stability, and R for Redundancy.

Table 5-2 shows a comparison between the participants' criteria for quality assessment of genomic data and our extended set of quality dimensions. Stars represent matches between experts (across columns) and dimensions (across rows). The last column indicates the total support for each dimension, based on the number of experts whose criteria matched the dimension. Some dimensions were grouped together because although we deem them different,

Table 5-2. Comparison between the experts' quality assessment criteria and our quality dimensions. Experts are the participants of our research study. Dimensions connected with a slash symbol are related and hence considered a single grouped dimension here. Stars along rows indicate an expert's criteria matching a quality dimension. The last column is the total support received by each dimension, based on the number of experts whose criteria matched the dimension. This is a summarized interpretation of the original textual responses shown in Table B-3 of Appendix B.

	Expert's identifier																
Dimension	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	Support
Density	*	*	*	*		*	*	*	*				*	*	*		11
Features	*	*	*	*			*	*	*				*	*	*		10
Publications / Literature Links	*		*			*		*				*			*		6
Gene Links			*											*	*		3
Structure Links			*											*			2
Other Links	*							*				*		*			4
Stability / Age / Freshness	*				*			*									3
Redundancy	*							*				*		*			4
Uncertainty				*				*		*	*					*	5

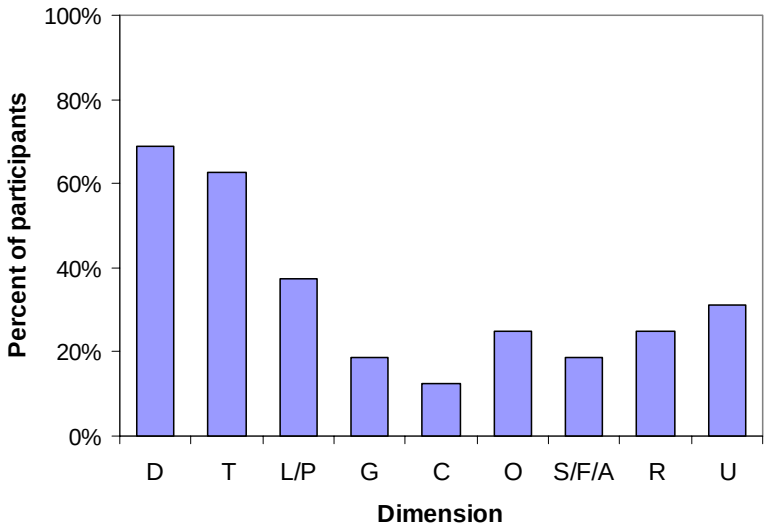


Figure 5-2. Percent of participants whose criteria for quality assessment of genomic data matched our chosen quality dimensions. D stands for Density, T for Features, L for Literature-Links, P for Publications, G for Gene-Links, C for Structure-Links, O for Other-Links, S for Stability, F for Freshness, A for Age, R for Redundancy, and U for Uncertainty.

they refer to similar quality characteristics of the data (for example, age, freshness, and stability are all temporal measures), and the criteria provided by the experts did not allow for a clear distinction between these dimensions. Also, it should be noted that the participants provided more criteria than the ones shown in this table, but we did not include them because they were not general enough, their computation could not be done efficiently, or they were subjective. Figure 5-2 presents the percent of expert support for each dimension.

The first dimension to include was Uncertainty, since it had good expert support (over 31%) and encompassed an aspect of quality that none of the other dimensions covered. Next, the Publications and Literature-Links dimensions (grouped in Table 5-2) were included as sub-dimensions of Density and Linkage, respectively, having over 37% expert support. They were treated as sub-dimensions because Publications is correlated with Density, and Literature-Links is correlated with Linkage. The problem with the Publications and Literature-Links dimensions is that conceptually they refer to the same aspect of quality (i.e., number of published works in the literature associated to a biological record), but in practice they need to be separated. Ideally, a count over the literature links of a record would suffice, but in reality such links are not always present, and furthermore, they can only link to literature databases within NCBI. Hence, we also need to count the number of textual references in a record to accurately measure the publication dimension. The reason why we do not drop the Literature-Links dimensions is because we (and biologists) see value in having a link as opposed to having simply a textual reference (see responses in Table B-10 from Appendix B). In Figure 5-2 we observe that the expert support for the joint Literature-Links / Publications dimension is about 37%.

Next, the Features dimension was included as a sub-dimension of Density, with an expert support of 62% (the highest among the new dimensions). As it was mentioned in Chapter 3, the

Features dimension refers to the amount of feature key annotations [21] in a genomic record, which are data of biological significance (clearly evident from the high expert support found). The two sub-dimensions added to Density (i.e., Features and Publications) are mainly unrelated to one another, but they both overlap with Density. We cannot drop Density because there are portions of the record information that are not accounted for by of the sub-dimensions. Besides, we believe Density has its own merit for being a general dimension that can be applied to atomic or complex data items within a record (not the case for its sub-dimensions), so it can give an estimate of the amount of information contained in a given part of the data tree (representing a record).

The next group of dimensions added to our set consisted of Gene-Links, Structure-Links, and Other-Links, all sub-dimensions of Linkage. As mentioned in Chapter 3, this division was made in order to exploit existing types of links, and was refined based on feedback gathered from expert collaborators at our university. From Figure 5-2 we observe that there is moderately good support for the Gene-Links (18%) and Other-Links (25%) dimensions, but only little support (12%) for the Structure-Links dimension. Nevertheless, we decided to keep the current division of Linkage dimensions until further experimental evidence was obtained either against or in favor of these dimensions.

Finally, the Age dimension which is grouped with the Freshness and Stability dimensions in Figure 5-2, was added to our set of quality dimensions because it had moderately good (18%) expert support, and we felt that this temporal aspect of quality was missing (it was certainly not well represented by either Freshness or Stability). As a final remark, we can see in Figure 5-2 that the initial dimensions have an expert support of at least 18%.

In summary, our initial set of five quality dimensions was extended to a total of twelve dimensions (including 6 sub-dimensions), based on findings from this experiment. The results from this experiment show that *i*) biologists deem our set of dimensions useful in assessing the quality of genomic data, and *ii*) biologists' own criteria for evaluating the quality of genomic data are partially matched by our dimensions; which we believe provide support for the biological relevancy of the chosen dimensions.

5.2.2.3 Experiment 2

The suitability of our set of quality dimensions was shown in Experiment 1. The next step is to evaluate the ability of the quality estimates (set of scores for all dimensions) to discriminate high versus low quality data. A related purpose is additionally sought: to find the key dimensions (or combinations of dimensions) for classifying data of high and low quality. A 2-point scale (with categories "high" and "low" quality) is used to make it less challenging for the experts to provide quality assessments and to reduce bias resulting from a finer scale (levels on a 5-point scale may have subjective interpretations).

Data Set. The *Expert* data set used for this experiment was directly obtained from domain experts through the research study we conducted (see Section 5.2.2), and consists of 371 records from the NCBI's Nucleotide and Protein databases, along with quality assessments for each of them (consisting of a category label: "good quality" or "poor quality"). A total of 187 records in this data set were classified by the study participants as having good quality, while a total of 184 were classified as having poor quality. We should note that although a total of 384 accession numbers (record identifiers) were collected in the study (see Tables B-1 and B-2 of Appendix B), 13 could not be included in the data set that was effectively used in the experiments. Reasons included repeated accession numbers, invalid accession numbers (did not correspond to any

record in the NCBI databases), and null scores for the Uncertainty dimension after loaded in the QMA (records without a sequence have a null uncertainty score).

Methodology. We loaded the data from the Expert data set into the prototype QMA, extracted the quality estimates, applied a logarithmic transformation to the scores that represented counts (Density, Features, Publications, Redundancy, and Linkage sub-dimensions), and standardized all the scores. To evaluate the ability of the quality estimates for discriminating data of high versus low quality, we first examined the distribution of quality scores for both the data classified as "good quality" (hereafter *high quality set*) and the data classified as "poor quality" (hereafter *low quality set*). Next, we built a binary classifier on top of our quality estimates and compared the predictions made by this classifier with the quality labels given by the experts. We describe the details of these steps next.

The logarithmic transformation was applied to reduce the influence of extreme values in the subsequent standardization process (computation of the mean and standard deviation), and to facilitate the inspection of the score distributions. The dimensions to which this transformation was applied (i.e., Density, Features, Publications, Redundancy, Literature-Links, Gene-Links, Structure-Links, and Other-Links) exhibited a prominent skew to the right of the distribution, which was lessen after the logarithmic transformation.

The inspection of the quality scores distributions for both the high and low quality sets is done using boxplots. For all quality dimensions, we show the boxplot of each quality set, side by side, with whiskers set at 3 times the interquartile range (IQR). Also, a statistical test is performed to test whether the distributions for the high and low quality sets are equal. The test used is the nonparametric Wilcoxon rank sum test, which tests the hypothesis of equal medians for two independent samples [52]. We chose this test over the two-sample *t* test primarily

because it makes no assumption about the distribution of the data (in particular, it does not assume normality, which does not hold for many of our dimensions' scores), and also because it is more robust to outliers in the data. Additional 3-dimensional plots are shown to further illustrate differences in the score distributions of the high and low quality sets, along selected dimensions.

The classifier used in this experiment was C4.5 [50], a well known and publicly-available decision tree classifier. We chose it because one can usually analyze its output (rules and decision tree) to gain insight into the classification problem at hand. C4.5 learns classification rules and builds a decision tree from a given training set; it then uses the generated decision tree to classify unseen test data. The *minobjs* parameter of C4.5, which specifies the minimum node size required to further partition a node, was set to 5 (default value) in all runs.

We used *k-fold cross-validation* [31, 59] (with k=10) to evaluate how well the model learned by the classifier predicts high and low quality data. In k-fold cross-validation, the data set is partitioned into k disjoint subsets of equal size. Then, the classifier is run k times, using one of the k subsets in turn as the test set and the remaining subsets as the training set. In this way, the classifier is tested on each of the k subsets exactly once, and the results from the k folds can be averaged to produce an estimator of the generalization error. In this experiment, the use of cross-validation was necessary due to the small size of our data set. For each of the 10 folds, we obtained the decision tree built by C4.5 over the training data, as well as the class-membership scores for both HQ and LQ classes. The decision trees were later analyzed to find the key dimensions for classification. The class-membership scores were used to make ROC curves and score plots, which support the discussion. Class-membership scores are the predicted posterior

probabilities for each class (HQ and LQ), and they were computed with Tanagra[51], a publicly available data mining software for academic and research purposes.

Results and Discussion. Figures 5-3 to 5-14 show the distribution of standardized scores for each quality dimension, over the high and low quality sets, using boxplots. We can observe that the distributions of the high and low quality sets for some dimensions have different central and dispersion tendencies (e.g., Figures 5-3, 5-4, 5-5, 5-6, and 5-8). In Figure 5-15, various three-dimensional plots of the quality scores for the high and low quality sets along selected dimensions are shown. Although these plots contain regions clearly dominated by one of the two quality classes, they also contain regions where data from both quality classes strongly overlap.

To quantify the observed differences in the medians of the two sets, we use the results of the Wilcoxon rank sum test (at the 5% significance level) shown in Table 5-3. In this table we observe that the Wilcoxon test's null hypothesis is rejected for five of our twelve dimensions: Density, Features, Freshness, Publications, and Stability; thus indicating that the high and low quality sets attain significantly different quality scores under these five dimensions.

The main problem faced when attempting to examine the discriminative capacity of our quality model's estimates is that although there is statistical evidence that the scores of some dimensions have different distributions for the high and low quality sets, there is still significant overlap between the distributions of the two quality classes, which makes it difficult to determine whether an individual dimension will be able to accurately discriminate high from low quality data.

To address the problem mentioned above, we used our quality estimates to build a classifier for the Expert data set, which was tested using a 10-fold cross-validation. The use of a decision tree classifier allowed us to evaluate the discriminating (or predictive) capability of our

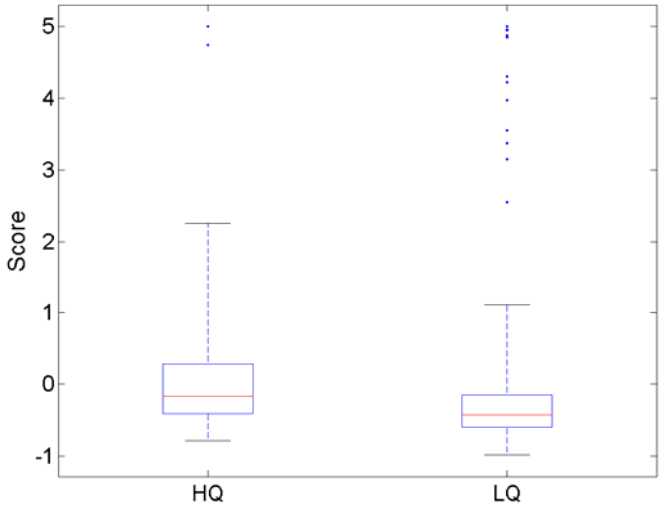


Figure 5-3. Distribution of standardized scores for the Density dimension over the Expert data set. The HQ boxplot shows the scores for the high quality records while the LQ boxplot shows scores for the low quality records. Boxplot whiskers are at 3 IQR.

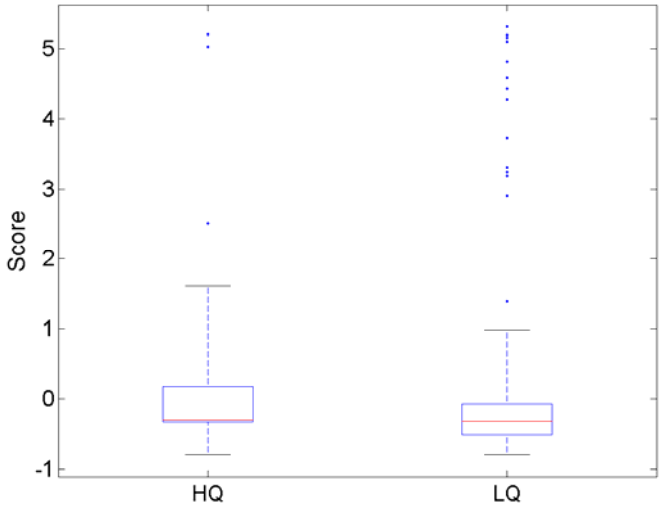


Figure 5-4. Distribution of standardized scores for the Features dimension over the Expert data set. The HQ boxplot shows the scores for the high quality records while the LQ boxplot shows scores for the low quality records. Boxplot whiskers are at 3 IQR.

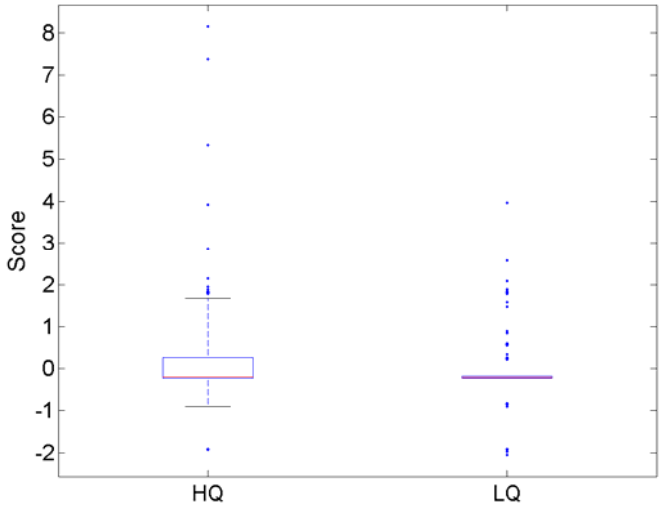


Figure 5-5. Distribution of standardized scores for the Publications dimension over the Expert data set. The HQ boxplot shows the scores for the high quality records while the LQ boxplot shows scores for the low quality records. Boxplot whiskers are at 3 IQR.

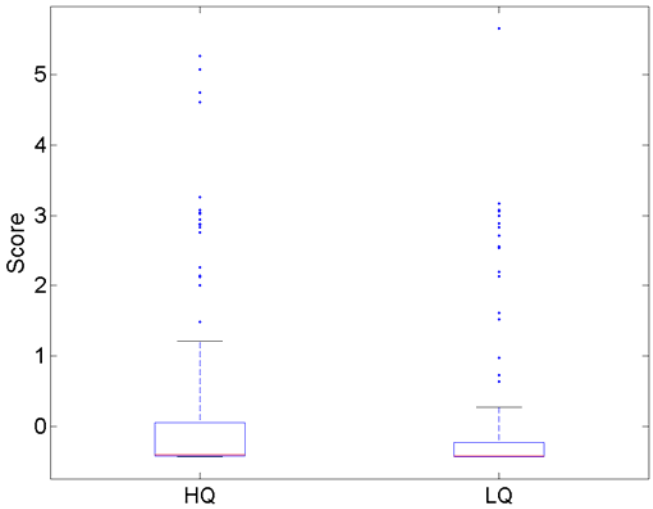


Figure 5-6. Distribution of standardized scores for the Freshness dimension over the Expert data set. The HQ boxplot shows the scores for the high quality records while the LQ boxplot shows scores for the low quality records. Boxplot whiskers are at 3 IQR.

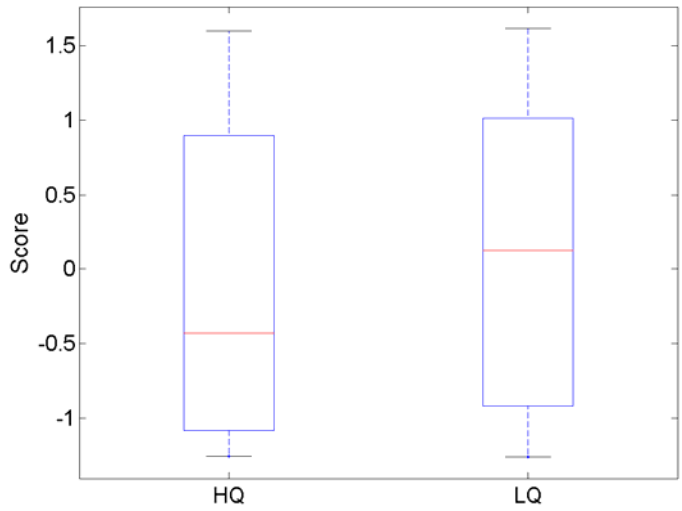


Figure 5-7. Distribution of standardized scores for the Age dimension over the Expert data set. The HQ boxplot shows the scores for the high quality records while the LQ boxplot shows scores for the low quality records. Boxplot whiskers are at 3 IQR.

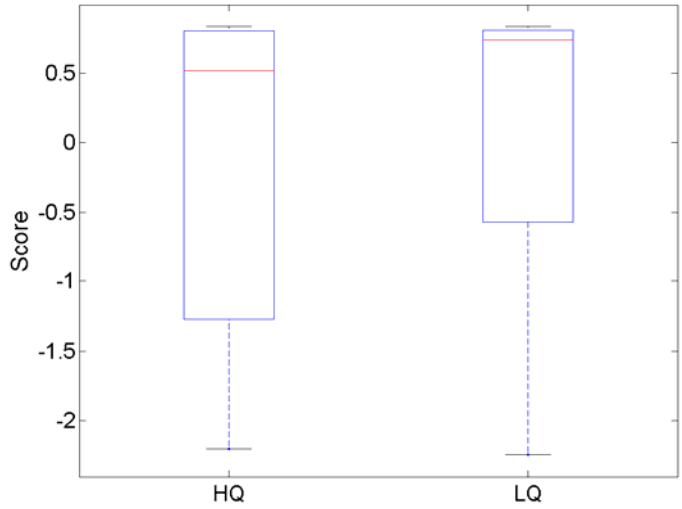


Figure 5-8. Distribution of standardized scores for the Stability dimension over the Expert data set. The HQ boxplot shows the scores for the high quality records while the LQ boxplot shows scores for the low quality records. Boxplot whiskers are at 3 IQR.

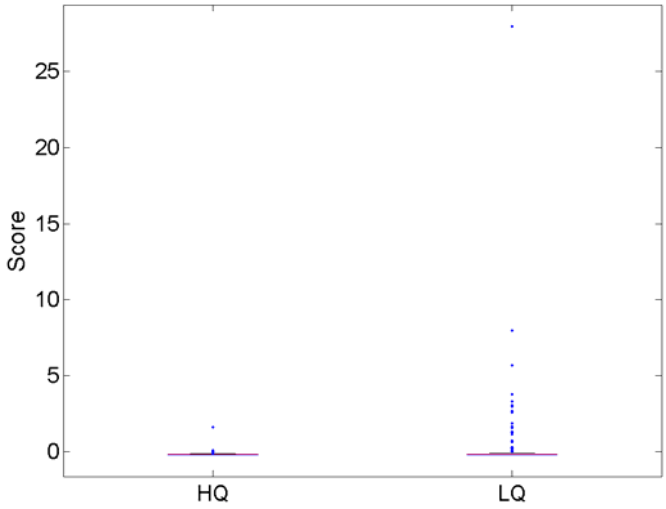


Figure 5-9. Distribution of standardized scores for the Uncertainty dimension over the Expert data set. The HQ boxplot shows the scores for the high quality records while the LQ boxplot shows scores for the low quality records. Boxplot whiskers are at 3 IQR.

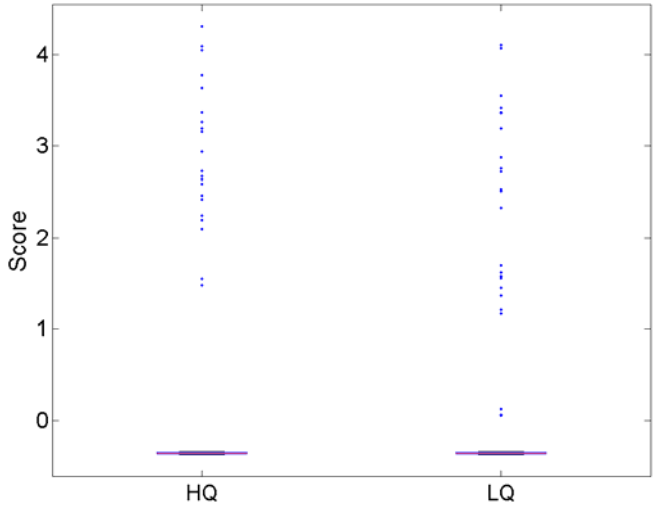


Figure 5-10. Distribution of standardized scores for the Redundancy dimension over the Expert data set. The HQ boxplot shows the scores for the high quality records while the LQ boxplot shows scores for the low quality records. Boxplot whiskers are at 3 IQR.

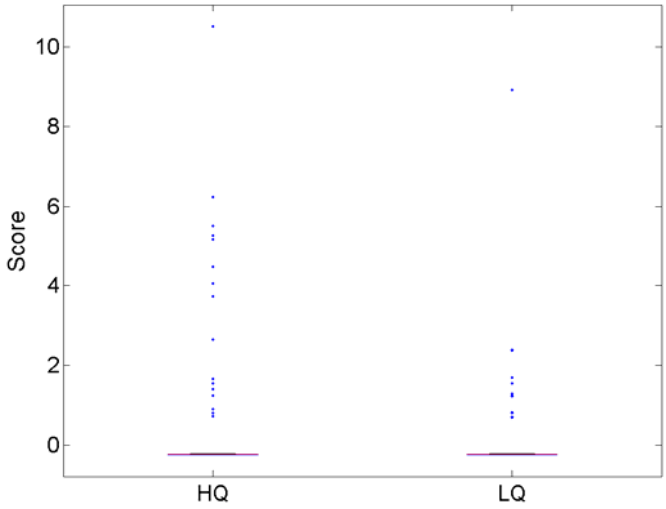


Figure 5-11. Distribution of standardized scores for the Literature-Links dimension over the Expert data set. The HQ boxplot shows the scores for the high quality records while the LQ boxplot shows scores for the low quality records. Boxplot whiskers are at 3 IQR.

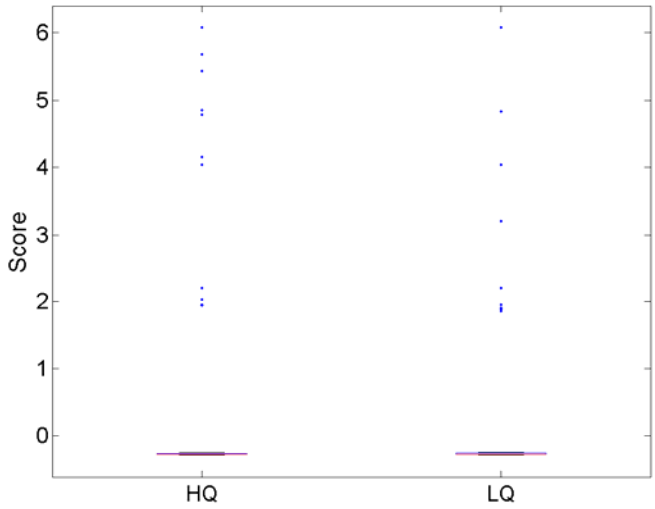


Figure 5-12. Distribution of standardized scores for the Gene-Links dimension over the Expert data set. The HQ boxplot shows the scores for the high quality records while the LQ boxplot shows scores for the low quality records. Boxplot whiskers are at 3 IQR.

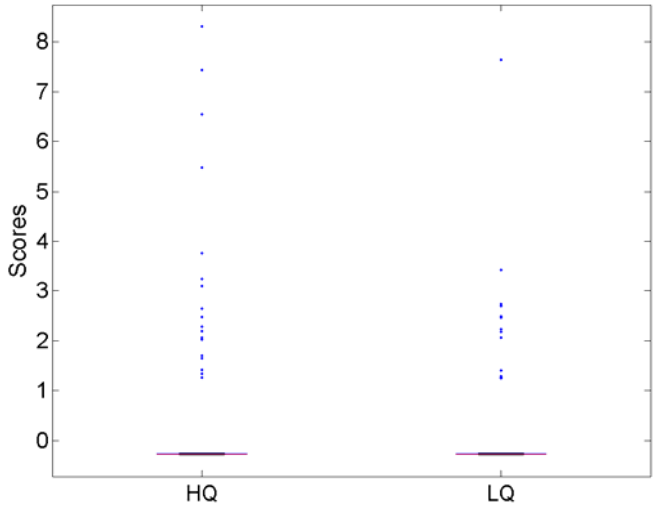


Figure 5-13. Distribution of standardized scores for the Structure-Links dimension over the Expert data set. The HQ boxplot shows the scores for the high quality records while the LQ boxplot shows scores for the low quality records. Boxplot whiskers are at 3 IQR.

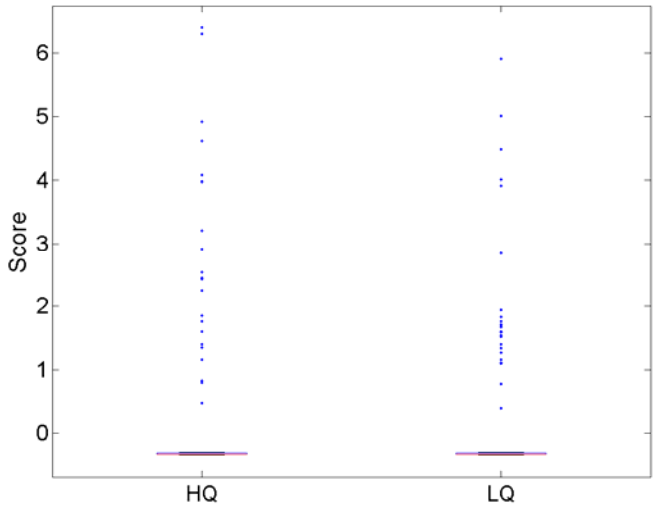


Figure 5-14. Distribution of standardized scores for the Other-Links dimension over the Expert data set. The HQ boxplot shows the scores for the high quality records while the LQ boxplot shows scores for the low quality records. Boxplot whiskers are at 3 IQR.

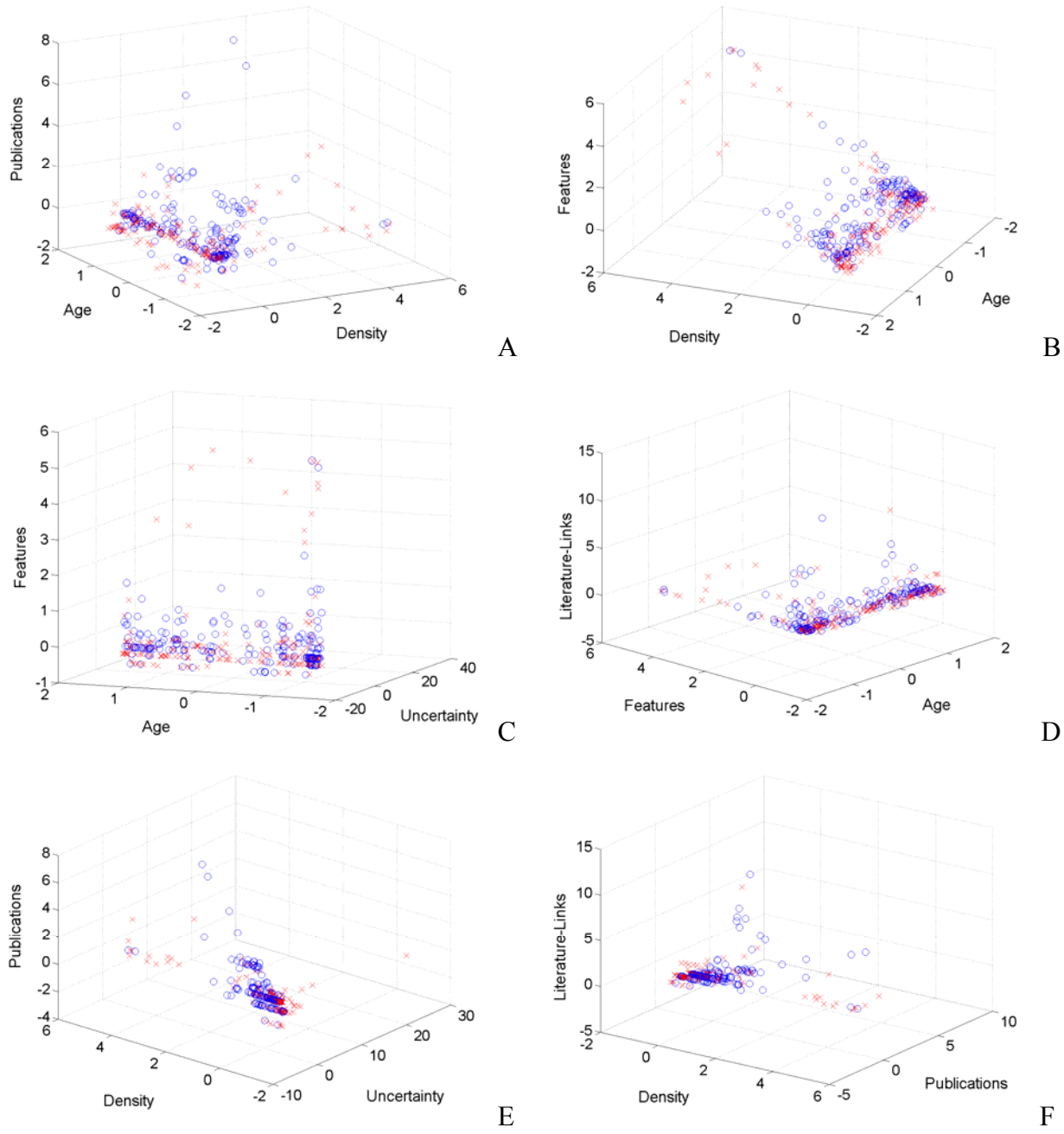


Figure 5-15. Three-dimensional view of the Expert data set along selected dimensions (scores are standardized). A) Plot along the Density, Age, and Publications dimensions. B) Plot along the Age, Density, and Features dimensions. C) Plot along the Uncertainty, Age, and Features dimensions. D) Plot along the Age, Features, and Literature-Links dimensions E) Plot along the Uncertainty, Density, and Publications dimensions. F) Plot along the Publications, Density, and Literature-Links dimensions.

Table 5-3. Wilcoxon rank sum test over the standardized scores for the high and low quality sets of the Expert data set. A 5% significance level was used for each of the quality dimensions. The median of each dimension's scores over the two data sets (HQ and LQ) is shown as reference. The last two columns contain the *p*-value and outcome of each test, respectively. The outcome indicates whether the null hypothesis H_0 was rejected. Results are sorted in ascending order of *p*-value.

Dimension	HQ median	LQ median	P-value	H_0 rejected
Density	-0.171	-0.429	0.000	yes
Features	-0.301	-0.321	0.000	yes
Freshness	-0.407	-0.420	0.030	yes
Publications	-0.205	-0.217	0.032	yes
Stability	0.520	0.741	0.035	yes
Uncertainty	-0.167	-0.167	0.052	no
Age	-0.430	0.128	0.100	no
Redundancy	-0.356	-0.356	0.476	no
Other Links	-0.326	-0.326	0.660	no
Gene Links	-0.276	-0.276	0.892	no
Literature Links	-0.236	-0.236	0.920	no
Structure Links	-0.273	-0.273	0.998	no

Table 5-4. Classifier performance over the Expert data set using a 10-fold cross-validation.

Fold	Generalization error (%)	AUC
1	35.1	0.63
2	13.5	0.83
3	35.1	0.75
4	29.7	0.71
5	29.7	0.78
6	27.0	0.78
7	51.4	0.47
8	18.9	0.83
9	37.8	0.63
10	34.2	0.67
Mean	31.3	0.71

combined quality estimates. The classifier selects a combination of relevant dimensions for classifying the training data, where relevancy is based on the *information gain* measure [31].

Table 5-4 shows two performance measures for the classifier built in each fold of the cross-validation, namely the Generalization error and the area under the ROC curve (AUC). We

observe that the average generalization (prediction) error obtained with the C4.5 classifier across the 10 folds of cross-validation is 31.3%. This corresponds to an average prediction accuracy of about 69%. A similar result is found when averaging the AUC scores across the 10 cross-validation folds, which gives 0.71. We also notice that there is a large variance in the results from different folds. The two most extreme cases correspond to folds 2 and 7, which exhibit the smallest and largest error percents, respectively.

We also evaluated the classifier by analyzing the class-membership scores, which are the predicted posterior probabilities of both the HQ (high quality) and LQ (low quality) classes. Ideally, the HQ-class scores of records in the high quality set should be close to 1, indicating a high probability of membership to the HQ class. Likewise, the LQ-class scores of records in the low quality set should be close to 1, indicating a high probability of membership to the LQ class. These class-membership scores are relevant because they represent all of our quality scores in a single figure that can be directly used for the classification of high versus low quality data (by setting a cutoff at 0.5 to make the class prediction).

Figure 5-16 shows the class-membership plots for each of the 10 folds of the cross-validation, over the Expert data set. Each class-membership plot contains either the HQ-class scores (blue line with circles) or the LQ-class scores (magenta line with crosses), depending on what the true class of the data samples is, and has the scores sorted in descending order (from left to right). When class-scores are above the 0.5 line, the classifier correctly predicts the class; when class-scores are below the 0.5 line, the classifier makes a prediction error. Hence, in a class-membership plot, crossing of the 0.5 line and the score line occurs further to the right (or never occurs) when the classifier performance is good; whereas crossing occurs further to the left when the classifier performance is poor. Similarly, if we are interested in high-confidence

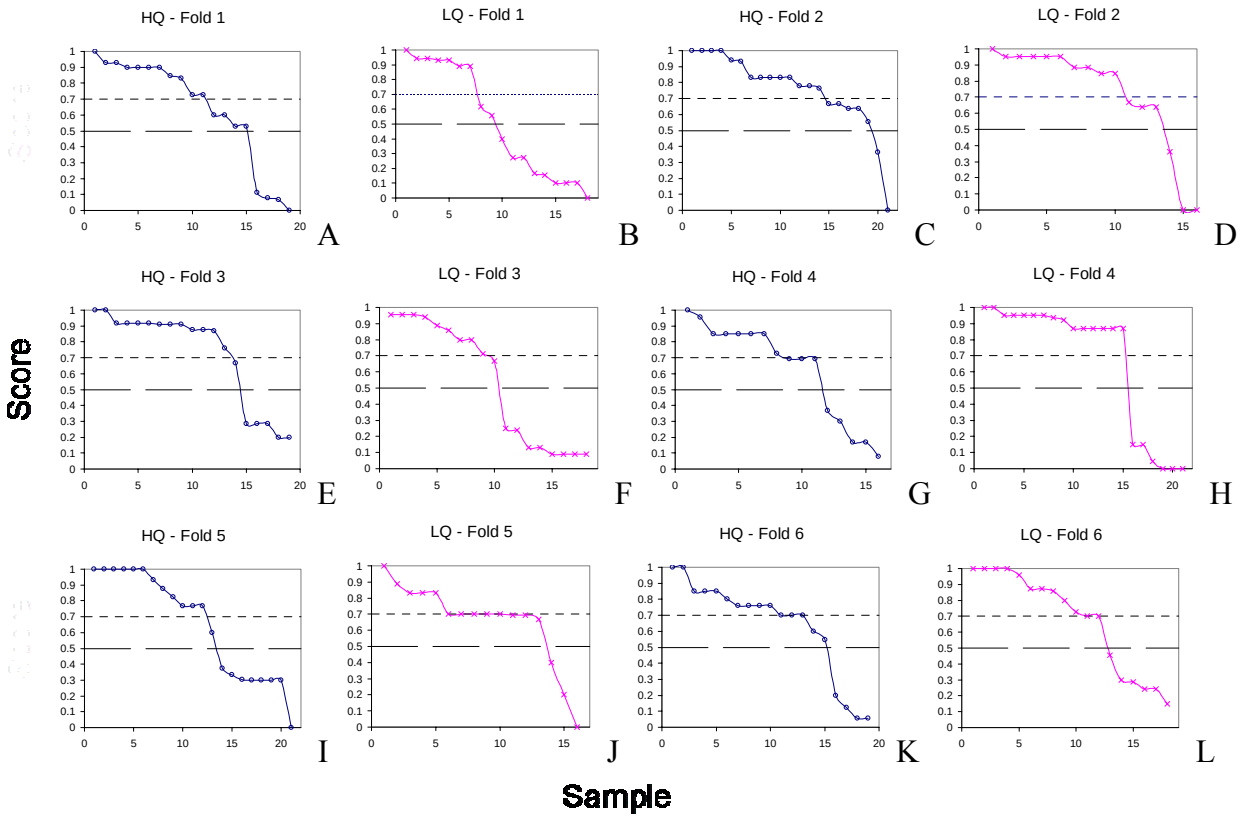


Figure 5-16. Class-membership scores for each fold of the cross-validation over the Expert data set. The blue line with circles represents HQ-class scores; the magenta line with crosses represents LQ-class scores. A) Scores for true HQ data in the test set of fold 1. B) Scores for true LQ data in the test set of fold 1. C) Scores for true HQ data in the test set of fold 2. D) Scores for true LQ data in the test set of fold 2. E) Scores for true HQ data in the test set of fold 3. F) Scores for true LQ data in the test set of fold 3. G) Scores for true HQ data in the test set of fold 4. H) Scores for true LQ data in the test set of fold 4. I) Scores for true HQ data in the test set of fold 5. J) Scores for true LQ data in the test set of fold 5. K) Scores for true HQ data in the test set of fold 6. L) Scores for true LQ data in the test set of fold 6. M) Scores for true HQ data in the test set of fold 7. N) Scores for true LQ data in the test set of fold 7. O) Scores for true HQ data in the test set of fold 8. P) Scores for true LQ data in the test set of fold 8. Q) Scores for true HQ data in the test set of fold 9. R) Scores for true LQ data in the test set of fold 9. S) Scores for true HQ data in the test set of fold 10. T) Scores for true LQ data in the test set of fold 10.

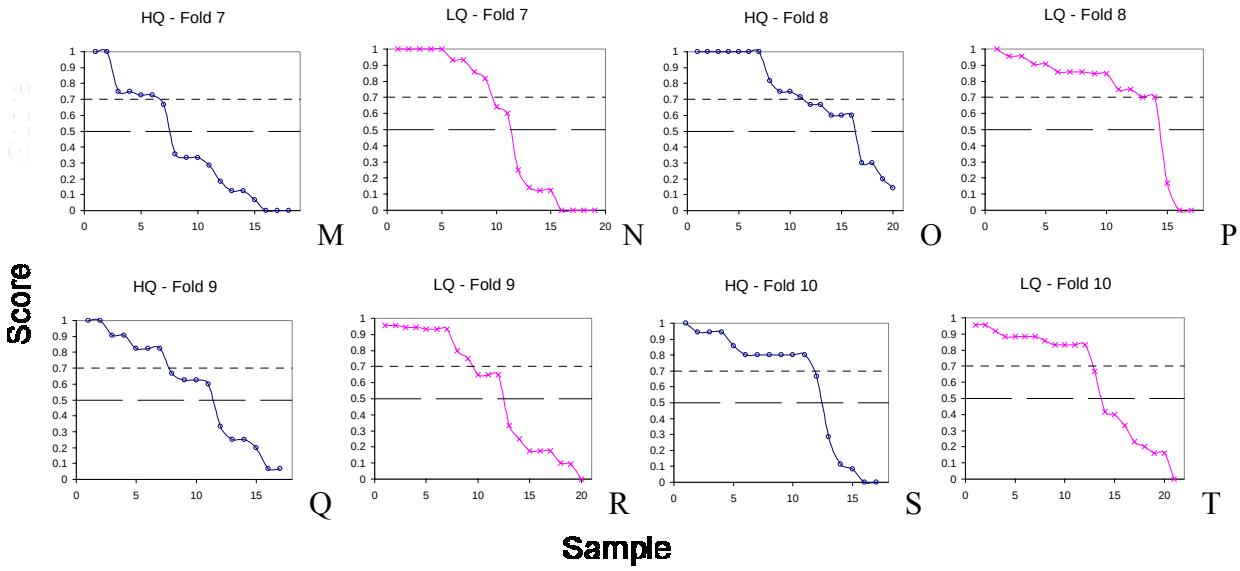


Figure 5-16. Continued.

Table 5-5. Classifier's prediction rate over the HQ and LQ sets of the Expert data set. Results are shown for each fold of the cross-validation, as percentages.

Fold	Prediction rate for HQ (%)	Prediction rate for LQ (%)
1	78.9	50.0
2	90.5	81.3
3	73.7	55.6
4	68.8	71.4
5	61.9	81.3
6	78.9	66.7
7	38.9	57.9
8	80.0	82.4
9	64.7	60.0
10	70.6	61.9
Total	71.1	66.3

predictions, class-scores should be compared against the 0.7 line, following the same principles outlined above. Table 5-5 shows the percent of correct predictions made by the classifier at each fold of the cross-validation for the high and low quality sets.

In the class-membership plots from Figure 5-16, we see that the classifier's predictive behavior varies moderately across folds. For example, in fold 7 the classifier has a prediction rate

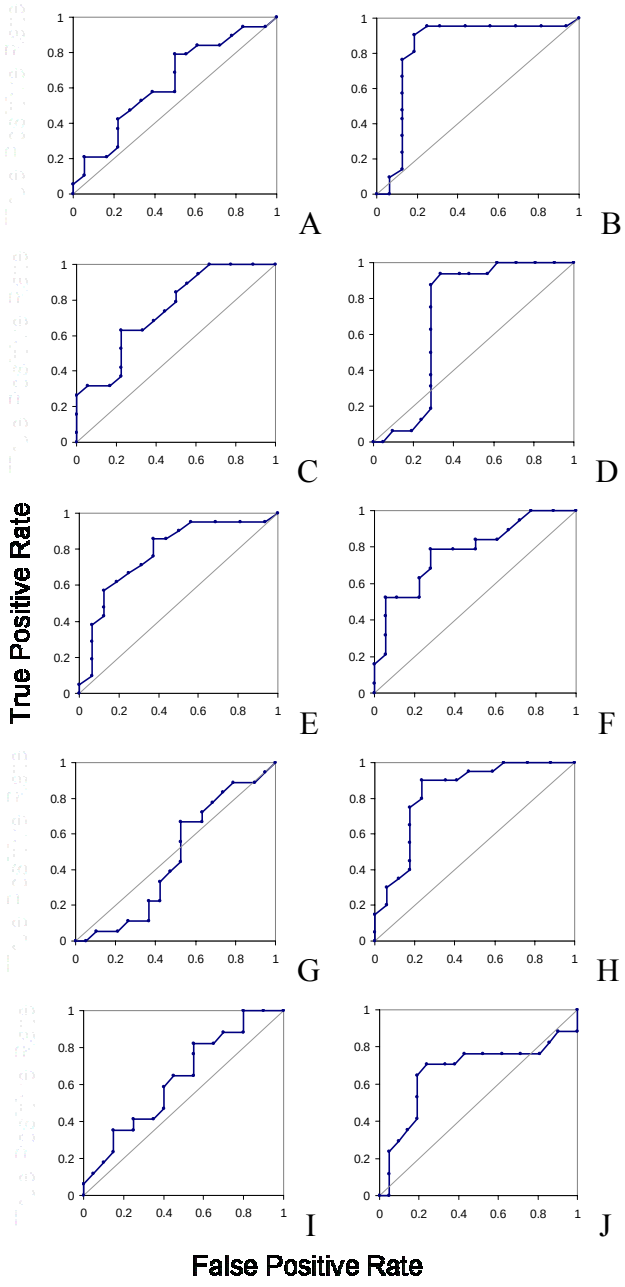


Figure 5-17. ROC curves for each fold of the cross-validation over the Expert data set. HQ is set as the positive class. A) ROC curve for test set in fold 1. B) ROC curve for test set in fold 2. C) ROC curve for test set in fold 3. D) ROC curve for test set in fold 4. E) ROC curve for test set in fold 5. F) ROC curve for test set in fold 6. G) ROC curve for test set in fold 7. H) ROC curve for test set in fold 8. I) ROC curve for test set in fold 9. J) ROC curve for test set in fold 10.

of less than 58% for both high and low quality sets (see Figure 5-16 parts M and N; and Table 5-5). On the other hand, in folds 2 and 4 the classifier has a prediction rate higher than 68% for both high and low quality sets (see Figure 5-16 parts C, D, G, H; and Table 5-5). We can also see marked differences in the confidence with which the classifier makes predictions (i.e., reflected in the magnitude of the class-score). For example, the classifier of fold 4 makes higher confidence predictions than the classifier of fold 6, especially for the low quality set (see Figure 5-16 parts G, H, K, L). Differences in the predictive behavior of the classifier across folds can also be observed in the ROC curves from Figure 5-17.

In Table 5-5 we also observe that the classifier correctly predicts more than 70% of the high quality data in 6 of the 10 folds, while it correctly predicts more than 70% of the low quality data in 4 folds. In average, the C4.5 classifier can correctly predict 71% of the high quality data and 66% of the low quality data. This is also illustrated in Figure 5-18, which shows the combined class-membership scores from all the cross-validation folds (union over disjoint test sets). From this figure, we observe that the point where the class-scores and the 0.5 line cross is located at approximately the 70^{th} percentile of the samples (towards the right of the plot).

We also explored was the use of different thresholds for the classification of our data set. The default threshold used by the classifier is 0.5 since the class-membership scores are actual probabilities. If we choose to use a higher threshold (e.g., 0.7) with the purpose of making only high-confidence predictions, it is possible that some data samples fall below that threshold (i.e., none of the class-scores is higher than 0.7). For such samples, we will assume that the classifier cannot confidently predict whether they belong to the HQ or LQ class (hence they will not be classified). The effect of different threshold values on the prediction rate and data selectivity (percent of the data that can be classified above the threshold) is shown in Table 5-6, for the high

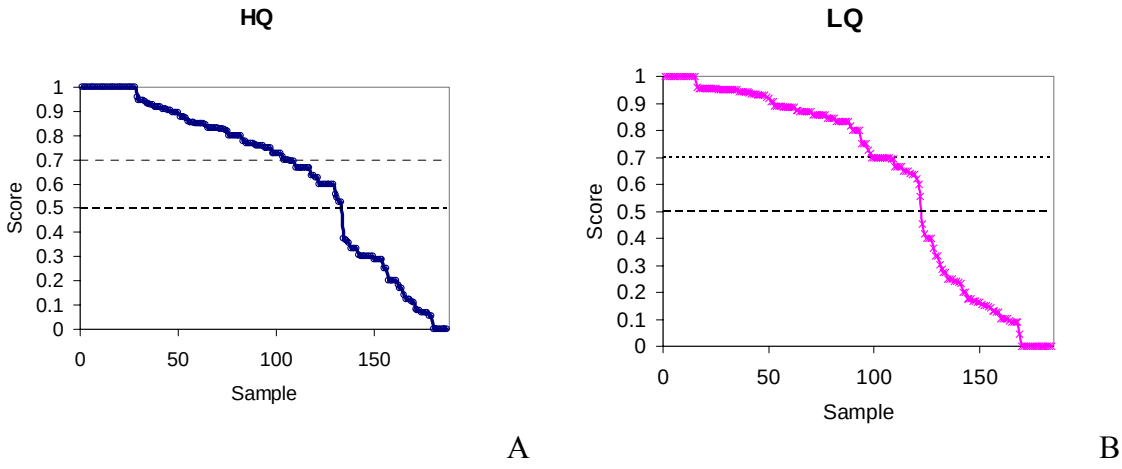


Figure 5-18. Class-membership scores across all cross-validation folds over the Expert data set. The blue line with circles represents the HQ-class scores; the magenta line with crosses represents the LQ-class scores. A) Scores for the high quality set (*true* HQ records, as labeled by experts). B) Scores for the low quality set (*true* LQ records, as labeled by experts).

Table 5-6. Effect of threshold value on the prediction rate and data selectivity over the Expert data set. Results are shown for the HQ and LQ data sets, and for the entire data set (last two columns).

Threshold	HQ data classified (%)	HQ prediction rate (%)	LQ data classified (%)	LQ prediction rate (%)	Total data classified (%)	Total prediction rate (%)
0.50	100.0	71.1	100.0	66.3	100.0	68.7
0.55	98.4	70.7	99.5	66.7	98.9	68.7
0.60	97.9	70.5	98.4	66.9	98.1	68.7
0.65	89.3	70.1	93.5	67.4	91.4	68.7
0.70	80.7	70.2	87.5	66.5	84.1	68.3
0.75	69.5	74.6	79.3	65.8	74.4	69.9
0.80	60.4	72.6	73.9	68.4	67.1	70.3
0.85	43.3	71.6	58.7	71.3	50.9	71.4
0.90	33.7	73.0	40.2	70.3	36.9	71.5

and low quality sets as well as for the entire data set. We can observe from this table that the prediction rate for the HQ and LQ sets does not exhibit a monotonic increase with higher threshold values. However, the overall prediction rate shows a roughly monotonic increase when

higher threshold values are used. Regarding data selectivity, we clearly see a decrease in the amount of data that can be classified when the threshold increases. However, it is interesting to note that is that when the threshold is increased up to 0.6, most of the data (98.1%) can still be classified, but as we further increase the threshold, the percent of data that is classified drops rapidly. The key finding from Table 5-6 is that the gain in prediction accuracy is only marginal compared to the loose in data selectivity, when the classification threshold is increased.

So far we established that it is possible to build a classifier based on all our quality estimates that distinguishes the expert-defined high and low quality sets with an accuracy of approximately 70%. Next we show that some of the dimensions in our original set of quality dimensions are key for the classification task at hand.

Table 5-7 shows the quality dimensions that are ranked among the top five split-attributes chosen by the decision tree classifier across the ten folds of the cross-validation. The second column of this table shows the number of supporting folds for each dimension (data in the table is sorted in descending order of supporting folds). A fold supports a dimension if the decision tree classifier built in this fold selected the dimension among its first five split-attributes. The third column of Table 5-7 shows the average rank obtained by each dimension, across the supporting folds. The rank of a dimension in a fold is defined by the order in which the dimension was selected by the decision tree classifier of that fold.

Table 5-7 therefore summarizes our analysis of the ten decision trees built in each of the cross-validation folds, regarding the most relevant dimensions for classification. The first seven quality dimensions shown in Table 5-7 (Uncertainty, Density, Age, Features, Literature-Links, Publications, and Other-Links) have support from more than half of the cross-validation folds, which makes them candidate key dimensions. The last two dimensions shown in Table 5-7 have

Table 5-7. Most relevant dimensions for classifying the HQ and LQ sets of the Expert data set. Dimensions ranked among the top five split-attributes selected by the decision trees across the 10 cross-validation folds are shown. Each row shows a dimension, the number of supporting folds for this dimension, and the average rank obtained by this dimension across its supporting folds.

Dimension	Number of supporting folds	Average rank
Uncertainty	10	1.0
Density	10	2.3
Age	10	3.9
Features	9	4.1
Literature Links	7	4.1
Publications	7	5.0
Other Links	5	3.2
Structure Links	2	3.5
Freshness	1	4.0

support from less than half of the folds, so we do not consider them as candidate key dimensions. The other three dimensions that are not present in Table 5-7 are dimensions that did not make it among the top-five split attributes selected by the classifier, in any of the folds. We proceed to analyze the seven candidate dimensions further.

The first three dimensions from Table 5-7 (Uncertainty, Density, and Age) are supported by all the cross-validation folds (i.e., by all decision trees). Remarkably, the Uncertainty dimension is selected as the first split-attribute in all ten trees, which is a very strong evidence of the usefulness of this dimension for classification. Density is selected as the second split-attribute in seven of the trees and as the third split-attribute in the remaining three trees, which is also strong evidence of its usefulness. Age is selected as the third or fourth split-attribute in eight of the trees, and as the fifth split-attribute in the remaining two trees. We can thus see a clear pattern of usage of these dimensions within the decision trees, in terms of which ones are used first in the classification process (clearly shown by the average rank for these dimensions in Table 5-7). Therefore, Uncertainty, Density, and Age become the initial set of key dimensions.

The next candidate dimension to consider is Features, the fourth dimension in Table 5-7. This dimension has support from nine of the ten folds, being selected as the fourth split-attribute in eight of the decision trees and as the fifth one in the remaining tree, for an average rank of 4.1. We thus have evidence for the usefulness of the Features dimension in the classification task (this dimension clearly becomes relevant after the top-three dimensions have been used), and we add it to our set of key dimensions. The next two dimensions from Table 5-7, Literature-Links and Publications, are both supported by seven folds, but differ in their average rank. Literature-Links is selected as the second split-attribute in two of the decision trees, and as the fifth split-attribute in remaining five trees, for an average rank of 4.1. Publications is selected as the fifth split-attribute in all seven trees, for an average rank of 5. Finally, the Other-Links dimension is supported by five folds, and has an average rank of 3.2.

Before deciding about the usefulness the last three dimensions, we tested the classifier's performance when each of these dimensions was given as input. For this purpose, we ran three different 10-fold cross-validations over the Expert data set using *i*) five dimensions: Uncertainty, Density, Age, Features, and Literature-Links, *ii*) six dimensions: Uncertainty, Density, Age, Features, Literature-Links, and Publications, and *iii*) seven dimensions: Uncertainty, Density, Age, Features, Literature-Links, Publications, and Other-Links. Table 5-8 shows the generalization error for three classifiers built over the Expert data set using different sets of candidate key dimensions. The results in Table 5-8 indicate that using more dimensions reduces the error. However, the reduction in the generalization error of the classifier when going from 6 to 7 dimensions is small compared to the reduction obtained when going from 5 to 6 dimensions. We thus decided to add both Literature-Links and Publications to our set of key dimensions since their inclusion as input attributes improved the classifier's generalization ability. On the other

Table 5-8. Comparison of the classification error obtained when using different sets of quality dimensions over the Expert data set. The generalization error of three classifiers built using 5, 6, and 7 candidate key dimensions (respectively) is shown as percentage.

	Number of dimensions used by classifier		
Fold	5	6	7
1	43.2	37.8	37.8
2	18.9	21.6	13.5
3	35.1	40.5	37.8
4	32.4	32.4	29.7
5	29.7	18.9	29.7
6	40.5	32.4	29.7
7	48.7	48.7	51.4
8	29.7	27.0	18.9
9	35.1	37.8	37.8
10	39.5	31.6	36.8
Mean	35.3	32.9	32.3
Stdev	8.3	8.9	10.7

hand, we decided to drop the Other-Links dimension from the set of candidates since its inclusion as input attribute did not improve much the classifier's generalization ability. In summary, our final set of key dimensions thus consists of Uncertainty, Density, Age, Features, Literature-Links, and Publications. The usefulness of these key dimensions for classifying high and low quality data is demonstrated by comparing the classifier's prediction error when using only the key dimensions versus using all the twelve dimensions (Tables 5-4 and 5-8). When using all the dimensions the average prediction error of the classifier is 31.3% (with standard deviation of 10.4%), meanwhile when using only the key dimensions the average prediction error of the decision tree classifier is 32.9% (with standard deviation of 8.9%).

5.2.2.4 Experiment 3

The previous experiment showed that a classifier based on all our quality estimates was able to distinguish (with 69% prediction accuracy) high versus low quality data labeled by experts. It also showed that similar results are obtained when using a smaller set of key

dimensions. The major drawback of this experiment was the use of a rather small data set (recall that one of our challenges was to obtain a reasonable large labeled data set from the experts). To address this problem we performed one more experiment, this time over a much larger data set with labels derived from experts' assessment of the overall quality of certain databases. The purpose of this experiment is to validate the usefulness of both the entire set of quality dimensions and the smaller set of key quality dimensions, for discriminating high from low quality data, over a large data set.

Data Sets. Three data sets were used for this experiment. We describe each of them next.

The *EST* data set consists of 780 records randomly sampled the NCBI's EST (Expressed Sequence Tags) database. The EST database contains only nucleotide records; specifically, single-pass cDNA sequences (or Expressed Sequence Tags), from a number of organisms [37]. The reason for choosing EST as the source database for one of our samples was because domain experts pointed out that most records in the EST database were of low quality. This was confirmed in our research study, as shown in Table 5-9, where the average rank given to the EST database indicates that it is regarded as a low quality database, at least compared to the other five databases considered. Hence, a "low quality" label was assigned to all records in our EST sample data set.

The *GenBank* data set consists of 780 records randomly sampled from the NCBI's GenBank database. The sample was specifically obtained from the CoreNucleotide division of GenBank , which essentially contains all nucleotide records that are not in the EST (Expressed Sequence Tags) or GSS (Genome Survey Sequence) databases. GenBank CoreNucleotide was chosen as the source database for one of our samples because the average rank given to this database by the study participants (see Table 5-9) indicated that it was regarded as the highest

Table 5-9. Participants' quality rankings for six NCBI databases. The last row shows the average rank of each database (lower ranks indicate higher quality). All other rows show the number of participants who assigned the rank in column one to the corresponding database. This is a summary of the original responses shown in Table B-4 of Appendix B.

Rank	dbEST	dbGSS	dbSTS	GenBank	HTC	RefSeq
1	0	1	0	9	1	6
2	5	2	2	4	2	2
3	1	2	2	2	2	3
4	2	3	0	0	0	0
5	2	0	1	1	1	0
6	0	1	0	0	0	2
N/A	6	7	11	0	10	3
Avg. rank	3.1	3.2	3.0	1.8	2.7	2.4

quality database, among the six presented. Hence, a "high quality" label was assigned to all records in our GenBank sample data set.

The *RefSeq* data set consists of 780 records randomly sampled from the NCBI's RefSeq database. Although RefSeq contains both nucleotide and protein sequences, only nucleotide records were considered for this sample with the purpose of maintaining consistency across data sets used in the experiments. RefSeq was chosen as the source database for one of our samples because domain experts indicated that most records from this database were of high quality (since this is a curated database). Results from our research study (see Table 5-9) confirmed that RefSeq is regarded as a high quality database (based on the average rank given to this database). Hence, a "high quality" label was assigned to all records in our RefSeq sample data set.

Methodology. We loaded the data from the EST, GenBank, and RefSeq data sets into the QMA, extracted the quality estimates corresponding to the key dimensions, applied a logarithmic transformation to scores representing counts (Density, Features, Publications, and Literature Links), and standardized these scores. Then, the ability of the key dimensions for discriminating data of high versus low quality was tested using *i*) the GenBank and EST data sets, and *ii*) the

RefSeq and EST datasets. These two pairs of data sets each includes one high quality set and one low quality set (the GenBank and RefSeq data sets are high quality sets –on average, while the EST data set is a low quality set –on average).

The steps involved in this experiment closely resemble the ones used in Experiment 2, namely building a binary classifier over the data set of interest, and measuring its discriminating capability (using the generalization error, AUC, ROC curves, and class-scores plots). However, the way in which we selected the test set for the classifier is different in this experiment: instead of using cross-validation, we separated 30% of the data (randomly chosen) for testing and used the remaining 70% for training. This could be done because the size of the data sets used in this experiment was much larger than the size of the Expert data set used in Experiment 2. The composition of the randomly selected GenBank-EST test set was: 238 high quality (HQ) records and 223 low quality (LQ) records. The composition of the randomly selected RefSeq-EST test set was: 237 high quality (HQ) records and 232 low quality (LQ) records. Additionally, we created some 3-dimensional plots using various combinations of key dimensions to show that high and low quality data were mostly separable.

Results and Discussion. We first discuss the results for the Genbank-EST data set. Table 5-10 shows the generalization error and AUC measures of performance for the classifier built over all dimensions and for the classifier built over the key dimensions only. These measures are computed over the GenBank-EST test data set. Figure 5-19A shows the ROC curve over the GenBank-EST test set, for the classifier built using all twelve quality dimensions. Figure 5-19B shows the ROC curve over the GenBank-EST test set, for the classifier built using only the six key quality dimensions. Figure 5-20 shows six three-dimensional views of the GenBank-EST test set, along select combinations of key dimensions.

Table 5-10. Classifier performance over the GenBank-EST data set. The generalization error and AUC measures are shown for the classifier built using all twelve quality dimensions, and for the classifier built using only the six key quality dimensions.

Dimensions	Generalization error (%)	AUC
All	0.64	0.999
Key	0.64	0.991

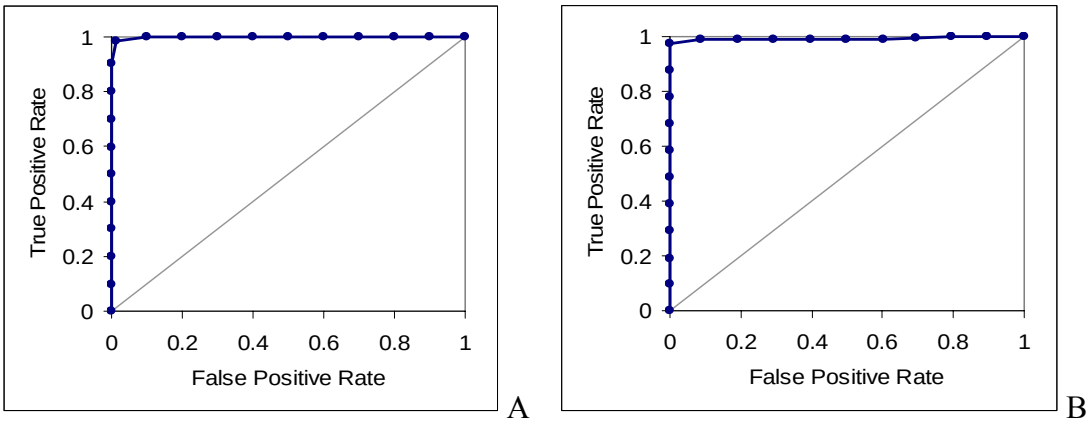


Figure 5-19. ROC curves of two classifiers over the GenBank-EST test set. HQ is set as the positive class. A) ROC curve for the classifier built using all twelve quality dimensions. B) ROC curve for the classifier built using the six key quality dimensions.

From Table 5-10 we observe that the generalization (prediction) error of the C4.5 classifier over the GenBank-EST data set is 0.6%, which corresponds to a prediction accuracy of 99.4%. A similar result is found with the AUC measure, which is close to 1 (see Table 5-10). Interestingly, the generalization error is the same for the classifier built over all the twelve dimensions than for the classifier built over the key dimensions. Likewise, the AUC values for these two classifiers are essentially the same. This can also be observed from the classifiers' ROC curves in Figure 5-19. Given that there is no significant difference in the predictive performance of the classifiers built over all dimensions and the classifier built over the key dimensions, we continue the experiment using only the key dimensions (plots in Figure 5-20 are based on key dimensions

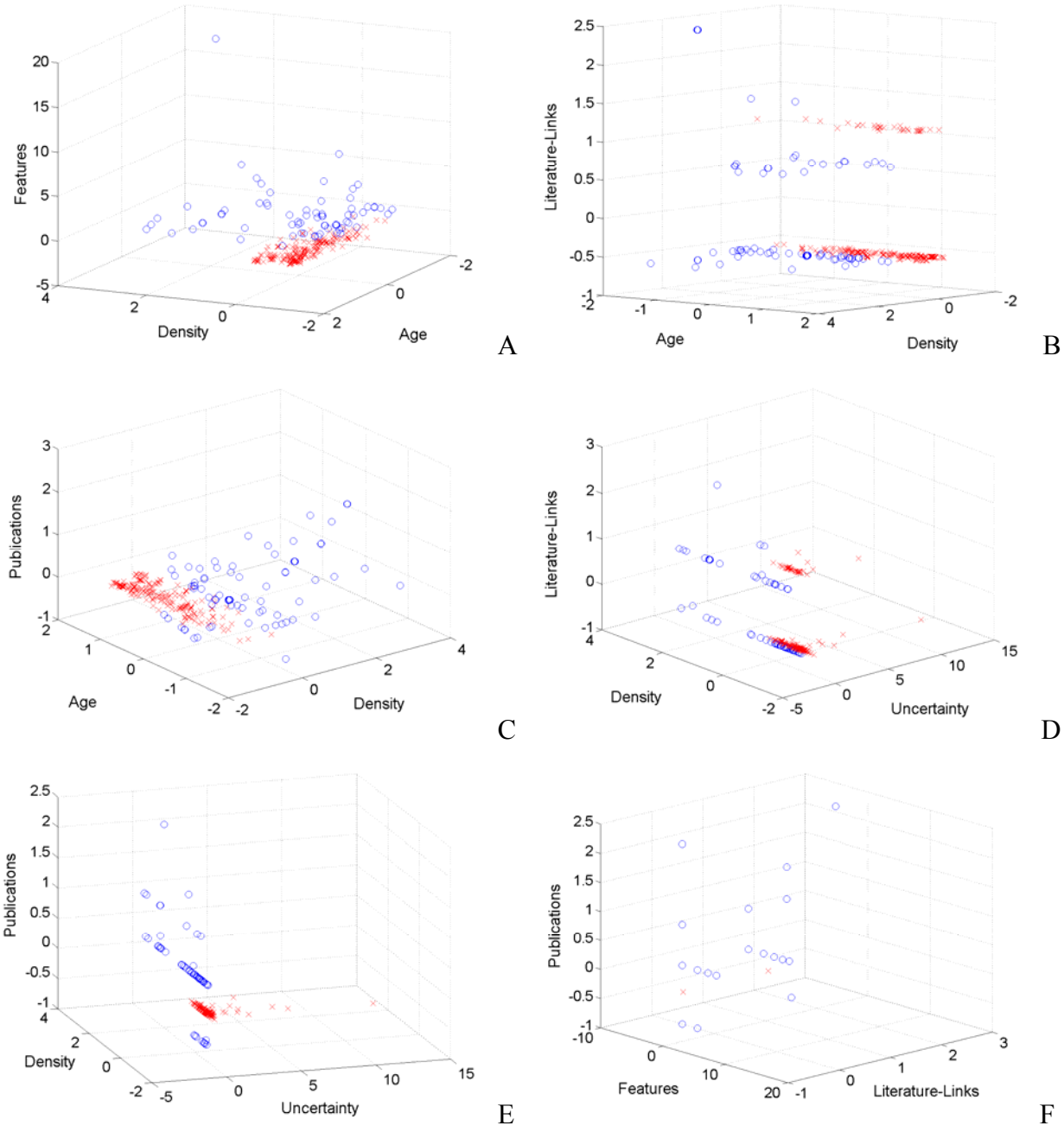


Figure 5-20. Three-dimensional view of the GenBank-EST test set along selected key dimensions (scores are standardized). A) Plot along the Density, Age, and Features dimensions. B) Plot along the Age, Density, and Literature-Links dimensions. C) Plot along the Age, Density, and Publications dimensions. D) Plot along the Density, Uncertainty, and Literature-Links dimensions E) Plot along the Density, Uncertainty, and Publications dimensions. F) Plot along the Features, Literature-Links, and Publications dimensions.

alone). In Figure 5-20 parts B, D, E, F we observe that the high quality data is clearly separable from the low quality data. In Figure 5-20 parts A and C we can see that there is a small overlap between the high and low quality data, but still they are mostly separable. Hence, combinations of key dimensions like the ones shown in Figure 5-20 are useful in separating the high quality from the low quality data.

We now discuss the results for the RefSeq-EST data set. Table 5-11 shows the generalization error and AUC measures of performance for the classifier built over all dimensions and for the classifier built over the key dimensions only. These measures are for the RefSeq-EST test data set. Figure 5-21A shows the ROC curve over the RefSeq-EST test set, for

Table 5-11. Classifier performance over the RefSeq-EST data set. The generalization error and AUC measures are shown for the classifier built using all twelve quality dimensions, and for the classifier built using only the six key quality dimensions.

Dimensions	Generalization error (%)	AUC
All	2.4	0.994
Key	0.4	0.999

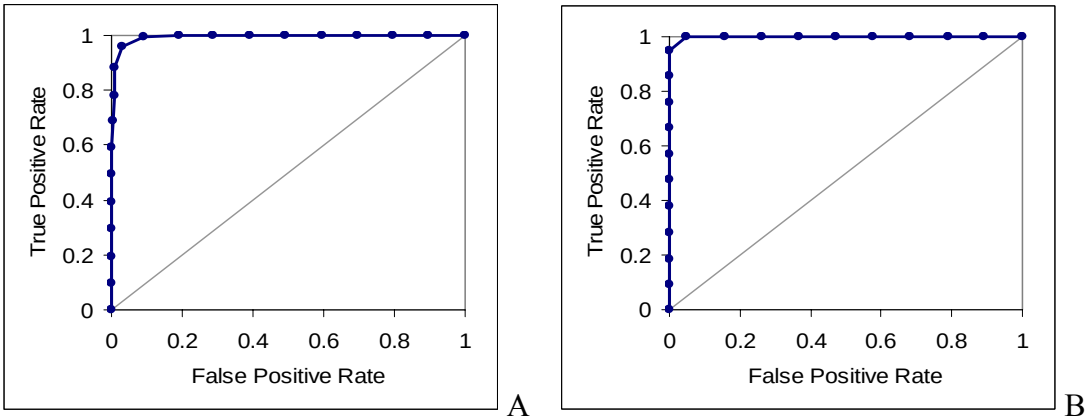


Figure 5-21. ROC curves of two classifiers over the RefSeq-EST test set. HQ is set as the positive class. A) ROC curve for the classifier built using all twelve quality dimensions. B) ROC curve for the classifier built using the six key quality dimensions.

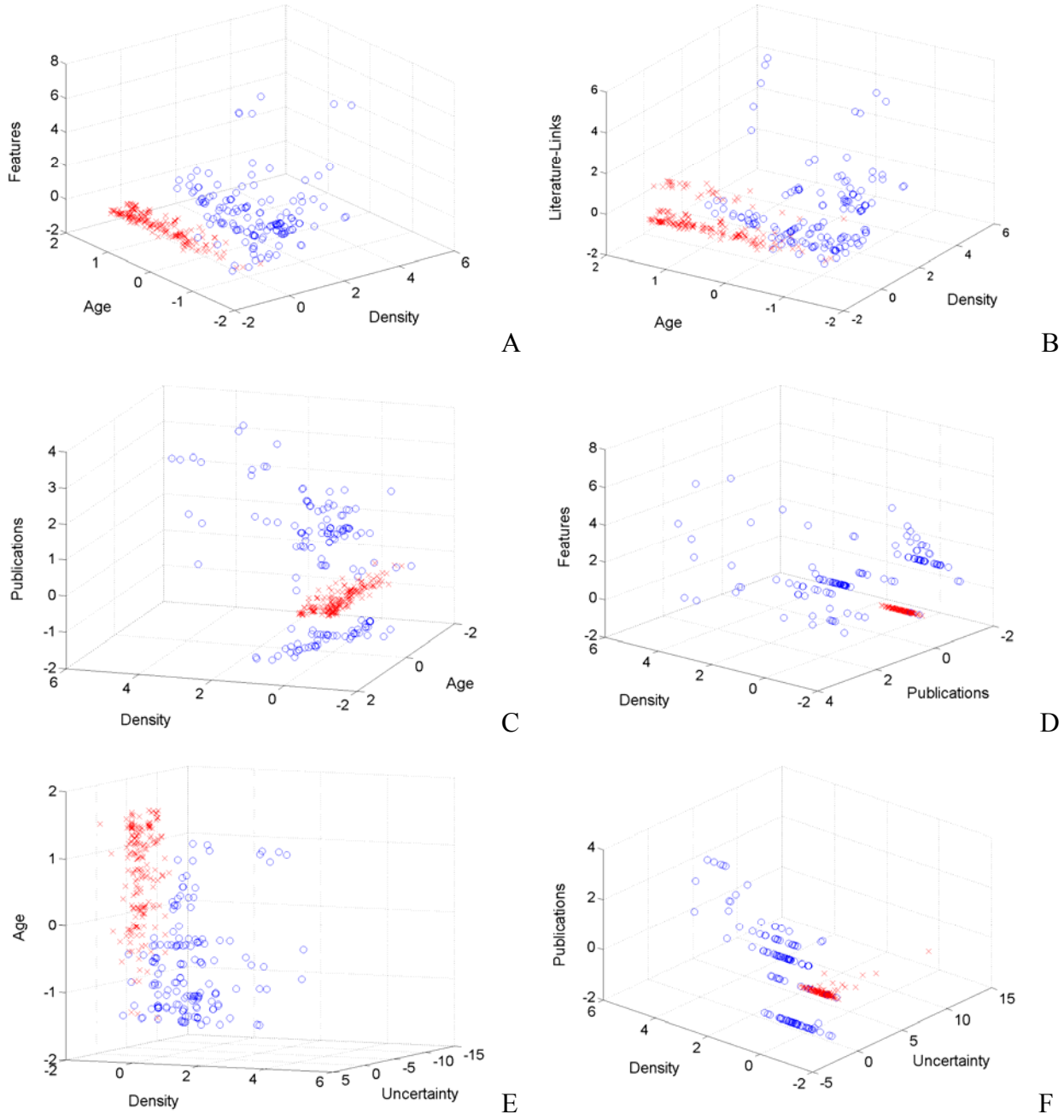


Figure 5-22. Three-dimensional view of the RefSeq-EST test set along selected key dimensions (scores are standardized). A) Plot along the Age, Density, and Features dimensions. B) Plot along the Age, Density, and Literature-Links dimensions. C) Plot along the Density, Age, and Publications dimensions. D) Plot along the Density, Publications, and Features dimensions E) Plot along the Density, Uncertainty, and Age dimensions. F) Plot along the Density, Uncertainty, and Publications dimensions.

the classifier built using all twelve quality dimensions. Figure 5-21B shows the ROC curve over the RefSeq-EST test set, for the classifier built using only the six key quality dimensions. Figure 5-22 shows six three-dimensional views of the RefSeq-EST test set, along select combinations of key dimensions.

From Table 5-11 we observe that the generalization error of the classifier built using all twelve dimensions is 2.4% (corresponding to a prediction accuracy of 97.6%), whereas the generalization error of the classifier built using the six key dimensions is 0.4% (corresponding to a prediction accuracy of 99.6%). An interesting finding is that the classification error is higher when all the twelve dimensions are used by the classifier than when only the key dimensions are used. Nonetheless, the AUC measures for the two classifiers are essentially the same (see Table 5-11), which can too be inferred from the ROC curves in Figure 5-21. Since the performance of the classifier built over all dimensions is at best comparable to the performance of the classifier built over the key dimensions, the remaining parts of the experiment were carried on using only the key dimensions. From the plots in Figure 5-22 we see that the high quality data is clearly separable from the low quality data (i.e., there is no major overlap among the two sets, for any of the plots). Hence, combinations of key dimensions like the ones shown in Figure 5-22 are useful in separating the high quality from the low quality data.

5.3 Evaluation of the Quality Management Architecture

As it was outlined in Chapter 4, the Quality Metadata Architecture (QMA) enables the integration of our Quality Estimation Model with existing biological repositories. The purpose of this evaluation is to highlight the most relevant capabilities of the QMA, and to validate the usefulness of our prototype QMA system based on its operational costs.

5.3.1 Relevant Capabilities of the QMA

5.3.1.1 Non-intrusive quality metadata augmentation

The QMA makes it possible to augment an existing biological data source with quality metadata in a non-intrusive way, by acting as a middleware layer between the original biological repository and the end users. One of the main advantages of the QMA is that it delivers its quality-aware services with minimal impact on the existing repository. By minimal impact we mean that no changes are made to the original data schema, format, or storage. The only changes attributable to the QMA are an increased query load for periodically polling data updates, and an enhanced quality-aware interface. The QMA makes this minimum-change feature possible by:

- Caching data from the existing repository into a local database, and enforcing coherency between the cache and the external data source.
- Providing an XML wrapper for the existing repository's native data format.
- Storing quality metadata in a separate repository from the data repository.
- Providing maintenance operations that refresh and update the quality information in the metadata source whenever changes to the data are detected.

We believe that the non-intrusive augmentation of quality metadata is a relevant feature of the QMA because it enables a smooth transition from current to quality-aware data sources. If the quality metadata augmentation was done more intrusively to trade in for performance, it is unlikely that current repositories would easily adopt such model, especially if many changes to the existing implementation are required.

5.3.1.2 Support for quality-aware queries

Although the User Interface (UI) is beyond the scope of the QMA, we still had to imagine how the UI would generally look like and especially what new functionalities would be needed in order to allow users to take full advantage of the quality information available. As a concrete example, let us take the NCBI's web interface system: Global Query. This UI allows general

text-based queries, queries by accession number (record identifier), and a variety of options or constraints for each of these queries. If we want to allow users of this system to have access to the underlying quality metadata (assume it has been already integrated into the NCBI system), two new functionalities are needed, namely *i*) to allow the user to choose if he wants to retrieve quality information along with the data being queried, and *ii*) to allow the user to choose which quality information to bring along with the data. A more refined functionality would additionally allow the user to specify how the quality information should be used during query processing; for example, the user could specify a filtering condition based on the value of one or more quality scores (our quality metadata is in the form of scores along various dimensions), or it could specify to sort (rank) the results based on a particular quality dimension. The addition of these new functionalities will convert the original interface into a quality-aware interface. All we need then is support from the QMA to do the actual query processing.

Inside the QMA, query processing is done by the Query Processor component. In the prototype QMA that we implemented, the Query Processor handles only queries based on accession number, but it supports both queries that retrieve only data and queries that retrieve data plus quality metadata. It also supports queries that retrieve a subset of the quality metadata by using an optional filter that indicates what dimensions to retrieve. We believe that enabling quality-aware queries is one of the most relevant capabilities of the QMA because it has a direct impact on both the way users interact with the biological data source, and the ability to succeed in obtaining the most valuable data for the task at hand.

5.3.2 Operational Cost for the Prototype QMA System

The operational costs of our prototype QMA system are divided into metadata retrieval and metadata computation costs. QM retrieval cost refers to the cost increase associated to processing

queries that involve quality metadata. QM computation cost refers to the cost increase associated to loading and maintaining the quality metadata in the QMA system.

Platform. The time measurements shown in this part of the evaluation were obtained using a 3.2GHz Pentium 4 machine with 2GB of physical memory and a single IDE disk, running Microsoft Windows XP Professional. The database system used was Oracle 10g Release 2 Enterprise Edition with buffer pool size set to 100MB. The Java Runtime Environment used was JRE 1.6.0.

5.3.2.1 Cost of metadata retrieval

We ran four different types of queries in our prototype QMA system: *i*) data-only queries, *ii*) data and metadata queries using all 12 dimensions, *iii*) data and metadata queries using 6 dimensions, *iv*) data and metadata queries using 1 dimension. The dimension used for the latter type of queries was Density, while the dimensions used for the 6-dimension queries were the "key" dimensions found in Experiment 2 (see Section 5.2.2.3), namely Uncertainty, Density, Age, Features, Literature-Links, and Publications. A total of 300 records were queried and the measurements for data and metadata retrieval time were averaged across records. We queried only records that were already loaded in the QMA system to avoid cache misses. Results of this experiment are shown in Table 5-12 and Figure 5-23.

Table 5-12 compares the retrieval time for data and quality metadata, per-record. Time measurements are performed for both record-level and node-level quality metadata, when using 1, 6, and 12 dimensions. Record-level quality metadata refers to the metadata associated to the root node of the XML tree (document) representing a record. Node-level quality metadata refers to the metadata associated to every node in the XML tree of a record. In Table 5-12 we observe that the cost of metadata retrieval is negligible with respect to the cost of data retrieval, even when using the node-level metadata. Particularly, the amount of time needed for retrieving the

Table 5-12. Per-record retrieval times for data and quality metadata. Time measurements (in milliseconds) are performed for both record-level and node-level quality metadata, when using different number of dimensions.

	QM - 12 dimensions		QM - 6 dimensions		QM - 1 dimension	
Data	Record	Node	Record	Node	Record	Node
2080	4.4	22.4	4.3	17.9	3.0	8.0

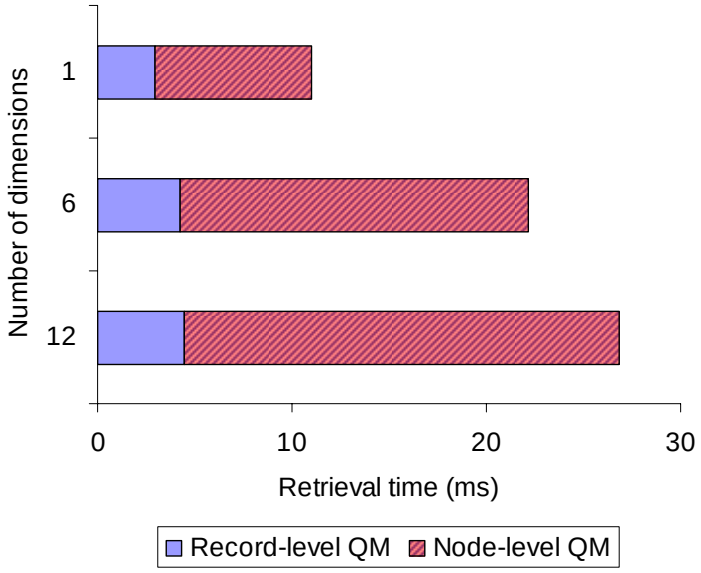


Figure 5-23. Retrieval time of quality metadata (per-record) for different number of dimensions. Time measurements (in milliseconds) are performed for both record-level and node-level quality metadata.

quality metadata is in average less than 1.1 % of the amount of time needed for retrieving the actual biological data. This proves that the increased cost of quality-aware queries (involving metadata retrieval) is small.

Figure 5-23 compares the per-record metadata retrieval time of queries involving 1, 6, and 12 quality dimensions, for both record-level and node-level quality metadata. This is an illustration of the data shown in Table 5-12 (data retrieval time is omitted). We can see in Figure 5-23 that the record-level QM retrieval time increases in 44% when going from 1 to 6

dimensions, while it only increases 14% when going from 6 to 12 dimensions. On the other hand, the node-level QM retrieval time clearly increases with the number of dimensions with a constant increase factor of 2.2 both when going from 1 to 6 dimensions and from 6 to 12 dimensions. Another observation is that the retrieval time of node-level metadata when using 1 dimension is 2.7 times larger than the retrieval time of record-level metadata; when using 6 dimensions it is 4.2 times larger, and when using 12 dimensions it is 5.1 times larger. In summary, we found that the cost for record-level quality metadata retrieval is small compared to the cost for node-level quality metadata retrieval, and that reducing the number of dimensions has significant savings in time when node-level metadata is requested.

5.3.2.2 Cost of metadata computation

Cost of bulk-loading. We bulk-loaded a total of 300 records from the NCBI databases into our prototype QMA system and measured the download time (i.e., data and link retrieval from the NCBI system) as well as the processing and storage time (i.e., computation and storage of quality metadata scores, as well as local caching of data). Bulk-loading in the context of our QMA system involves both data and metadata. However, for comparative purposes we bulk-loaded the system using only data (i.e., no metadata computation or storage was performed). Results of this experiment are shown in Table 5-13.

Table 5-13 compares the per-record bulk-load time of data and metadata with the bulk-load time of data-only. Bulk-load time is divided into two parts: download time, and processing plus storage time. In Table 5-13 we observe that when both data and metadata are bulk-loaded (versus data-only), the download time increases by a factor of 7, and the processing and storage time increases by a factor of 4. When the total bulk-load time is considered, there is an increase factor of 6 for data and metadata versus data only. Interestingly, we see that the download time is a significant part of the total bulk-load time, contributing with 71% to the data bulk-load time, and

Table 5-13. Per-record bulk-load times for data versus data and quality metadata. Time measurements (in milliseconds) for the two main parts of the bulk-load process (download, and processing plus storage) are shown.

	Data	Data and Metadata
Download	1844	12346
Processing and Storage	762	2752
Total	2605	15097

with 82% to the data plus metadata bulk-load time. We believe the download time could be greatly reduced if we were not limited by the "3-seconds rule" imposed by the NCBI administrators. This rule restricts the frequency of queries to the NCBI databases to no more than 1 every 3 seconds.

Although the cost of data and metadata bulk-loading is significantly large (15 seconds per record) with respect to the cost of data-only bulk-loading (2.6 seconds per record), this is a one-time-only cost that we will run into only when the existing biological repository is migrated for the first time to the quality-aware system enabled by the QMA.

Cost of maintenance. A total of 57 records were used, for which updates (i.e., multiple versions) were available. These records were first loaded into the prototype QMA system. Then, maintenance cost was measured for subsequent versions in terms of download time (data and link retrieval from NCBI) as well as processing and storage time (update and storage of quality metadata and data). In the context of our QMA system, maintenance involves both data and metadata; however, for comparative purposes we updated the system using only data (i.e., simply deleting the previous version of the record and storing the newer one into the data cache). Results of this experiment are shown in Table 5-14.

Table 5-14 compares the maintenance times for data and metadata per-record. Maintenance time is divided into two parts: download time, and processing plus storage time. The results in

Table 5-14. Per-record maintenance times for data versus data and quality metadata. Time measurements (in milliseconds) for two parts of the maintenance process (download, and processing plus storage) are shown.

	Data	Data and Metadata
Download	15	10620
Processing and Storage	2544	4603
Total	2560	15223

Table 5-14 show that when metadata maintenance is performed, the download time increases by a factor of approximately 700 with respect to data-only maintenance. On the other hand, the processing and storage time increases only by a factor of approximately 2 for metadata maintenance. The total maintenance time increases by a factor of 6 for data and metadata versus data only. This cost increase is similar to the one previously obtained for bulk-loading, and the reason is that the link-download step that is common to both maintenance and bulk-load processes has a large cost. This step accounts, in average, for 70% of the maintenance time, and for 68% of the bulk-load time, when metadata is involved. During maintenance, the data-download time is actually negligible with respect to the link-download because data updates are downloaded from the NCBI ftp site in a compressed format, which allows for a low per-record cost. We restate here that the download time is negatively affected by the practical limitation imposed by NCBI regarding query frequency (see description of the "3-seconds rule" in previous section).

It is important to mention that the results of this experiment are based on the atypical scenario that all records in the cache are updated on the same day. Usually, we expect only a small fraction of the cached records to be updated on any single day, which would have the effect of reducing the overall maintenance cost. For GenBank and RefSeq, the percent of records

that are daily updated is about 0.05% and 0.03%, respectively[15]. Assuming the same update rate for records in the QMA, the amortized maintenance cost per-record is 1.2 seconds, which is 13 times smaller than the cost suggested in Table 5-14.

[15] This estimate was obtained using five daily updates downloaded in the month of October, 2007.

CHAPTER 6
CONCLUSIONS AND FUTURE WORK

6.1 Conclusions

We developed a model for estimating the quality of data in biological databases. This model defines various quality dimensions and their respective measures. Unlike previous works in the area, our model is based on quality dimensions that can be quantitatively measured using data already stored in data sources. Part of the novelty of our approach resides in the way our quality estimates are systematically computed for biological data represented in a semi-structured model.

Along with the quality estimation model, we developed the quality management architecture, which enables the integration of the quality model into an existing biological database. The key principle of this architecture is to minimize the changes to the existing system, which we believe is necessary for a smooth transition from current to quality-aware data sources.

The biological significance of the Quality Estimation Model was evaluated using expert-feedback, gathered through a research study that we conducted among 16 biologists. The usefulness of the generated quality estimates to discriminate high versus low quality data was experimentally tested using various data sets from the NCBI databases, for which quality assessments were either directly obtained from experts or derived from general ratings of the source databases by the experts. Results of this evaluation show that it is possible to build a classifier, based on our quality estimates, that distinguishes high quality from low quality data (labeled by experts) with a prediction accuracy of 69%. Results from the semi-automatically labeled data sets show that the high quality and low quality data are clearly separable, which is evident from the prediction accuracy of 0.6% and 0.4% obtained by the classifiers built over the GenBank-EST and RefSeq-EST data sets, respectively.

The usefulness of the Quality Management Architecture was also evaluated with respect to its functional capabilities and operational costs (for query, bulk-load, and maintenance operations), using the implemented QMA prototype. A subset of the NCBI databases was used in this case as the biological repository to be integrated with the QEM through the QMA. This evaluation shows that two of the most relevant and beneficial capabilities of the QMA are the non-intrusive quality augmentation, and the support for quality-aware queries. Evaluation results of the operational costs of the QMA show that the increased cost of metadata retrieval (in queries) is less than 1.1 % of the cost for data retrieval. For bulk-loading and maintenance costs, there is an increase factor of 6 for data and metadata versus data only.

6.2 Contributions

We believe our work represents an important contribution to the area of Data Quality in the context of biological databases. First, we identified a set of quality dimensions that are objective, measurable, and biologically-relevant. Second, for each quality dimension, we formulated a quantitative measure that can be computed when data is represented in a semistructured model. Third, we defined a core set of maintenance and query operations for managing the quality metadata associated to the biological data, under a semistructured data model. Fourth, we designed and implemented a quality management architecture that enables the integration of our quality model (dimensions, measures, and operations) with existing biological repositories. Fifth, we conducted a research study among sixteen domain experts (biologists) with the purpose of collecting experts' assessments about the quality of genomic records, which were used during the evaluation of our quality model. Sixth, we performed an experimental evaluation of our quality model using the responses collected from our research study, as well as data from the NCBI's databases (a public genomics repository widely used by the scientific community). Results from this evaluation show that our set of dimensions are useful in assessing the quality of genomic

data, and that our quality estimates can be used to build a classifier that discriminates high quality data from low quality data. Additionally, we evaluated the usefulness of the prototype quality management architecture that was implemented by showing that a smooth migration from current to quality-aware data sources will be facilitated by the QMA, and that users of current data sources will benefit from the quality-aware queries enabled by the QMA.

6.3 Future Work

We now outline some possible directions for future work. One area in which we foresee future work is the development of benchmarks for biological data quality. In fact, the lack of such benchmarks was one of the main challenges we faced during the evaluation phase of our work. We overcame it by means of a research study conducted among expert biologists, but this study was limited both in the number of participants (only 16 experts) and in the number of quality assessments that were collected (only 24 per expert). We believe more studies like this are needed so that more test data is available to future researchers.

A second future research direction is the exploration of new cache replacement strategies that are optimized for the typical access pattern of biological data. Example queries or sequences of queries (e.g., data workflow) would need to be collected from users of current repositories in order to design query profiles that help design cache algorithms. We believe that the link structure of biological records from sources like NCBI could be exploited to either pre-fetch or evict data in the cache. As a pre-fetching strategy, data linked to already cached data can be loaded into the cache before it is actually requested by a user. The hypothesis here is that users normally expand their initial set of results by searching data linked from the initial set. As a victim selection strategy, link information can be used to evict the record with least number of links (the hypothesis here being that highly-linked records are more likely to be accessed in the future, so it is better to keep them in the cache as long as possible).

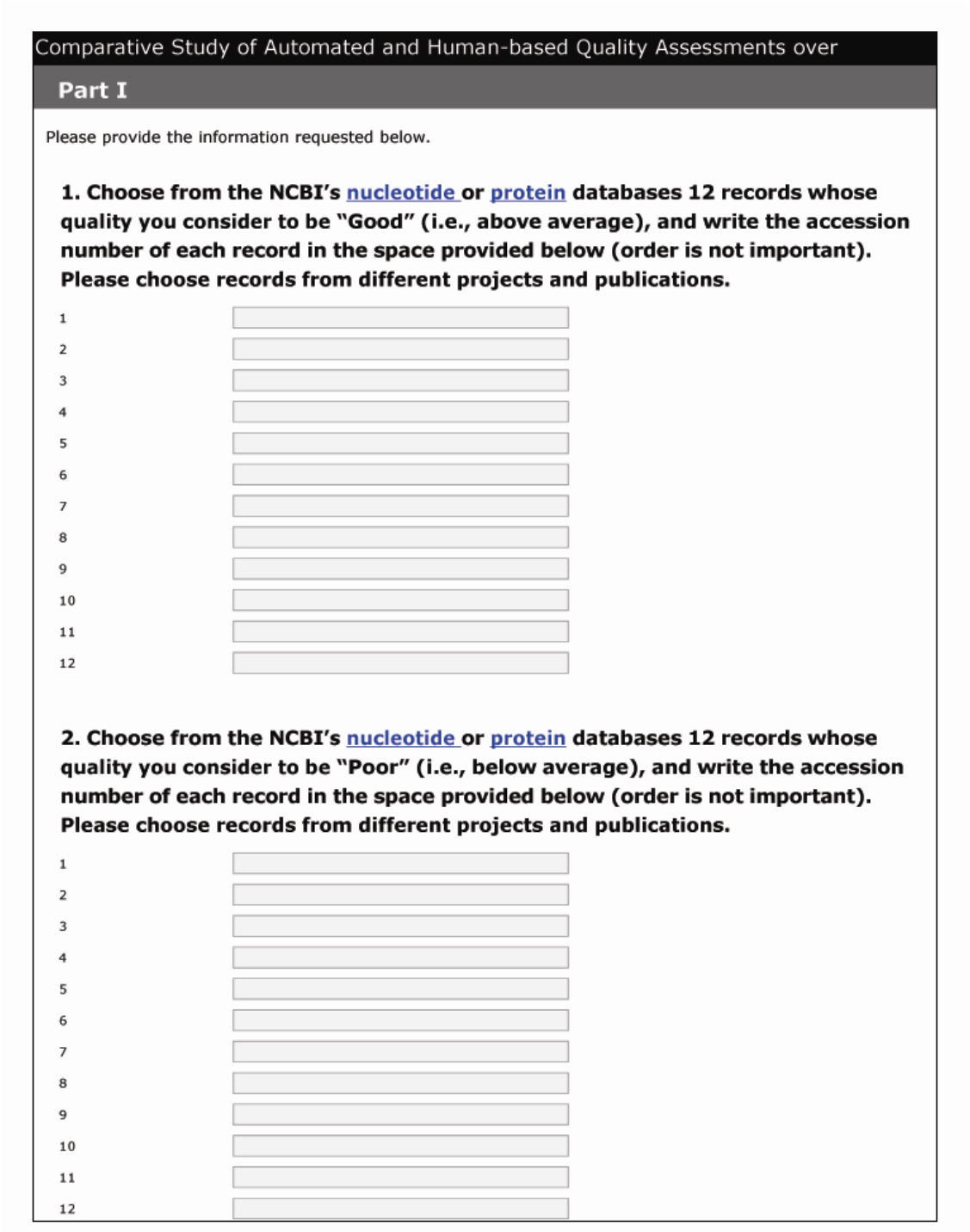
Future research could also be directed towards the improvement of the current measures and/or expansion of the current set of quality dimensions. More efficient metrics and operations could be investigated, as well as the incorporation of new dimensions based on special information offered by different biological repositories. Granularity levels at which quality information is relevant could also be investigated (e.g., record-level may require different quality dimensions and measures than database-level or sub-record level).

Finally, we envision future research in aspects related to the management of quality. In particular, our quality management architecture could be improved by the development of a storage-optimization strategy that exploits the hierarchical structure of the data (in a semistructured representation) to inherit quality metadata. Savings in metadata storage and maintenance could be significant if many child nodes have the same quality scores of their parent node.

APPENDIX A
SURVEY QUESTIONNAIRE

Online Survey Questionnaire presented to the participants of the research study

"Comparative Study of Automated and Human-based Quality Assessments over Genomic Data".



Comparative Study of Automated and Human-based Quality Assessments over

Part I

Please provide the information requested below.

1. Choose from the NCBI's nucleotide or protein databases 12 records whose quality you consider to be "Good" (i.e., above average), and write the accession number of each record in the space provided below (order is not important). Please choose records from different projects and publications.

1
2
3
4
5
6
7
8
9
10
11
12

2. Choose from the NCBI's nucleotide or protein databases 12 records whose quality you consider to be "Poor" (i.e., below average), and write the accession number of each record in the space provided below (order is not important). Please choose records from different projects and publications.

1
2
3
4
5
6
7
8
9
10
11
12

Comparative Study of Automated and Human-based Quality Assessments over

Part II

Please answer each of the following questions to the best of your knowledge.

1. What criteria do you use when evaluating the quality of a genomic record? If you include several criterions, please rank them in order of importance (1 being the most important, 2 being the second most important, etc). Also, please include a description of each quality factor you provide.

2. How would you rank the overall quality of the following NCBI data sources? Use ranks from 1 (highest quality) to 6 (lowest quality), or N/A (if you have never used the database). You may use the same rank for two or more databases if you consider them to be equivalent in terms of their quality.

	Ranking
dbEST	
dbGSS	
dbSTS	
GenBank	
HTC	
RefSeq	

3. When using genomics databases, do you usually work with a well defined set of records on a regular basis or do you work with a different set of records every time? Explain briefly.

4. Do you know of any genomics database that provides quality scores for its data? If so, please mention it here.

Comparative Study of Automated and Human-based Quality Assessments over

Part III

Evaluate the usefulness of the following factors in assessing the quality of genomic records. For each factor, select a category and briefly justify your choice. Also, feel free to add any comments relevant to how these factors may be enhanced or extended.

1. STABILITY

Description: A record is stable if its contents (both sequence and annotations) remain mostly unchanged during various database updates. Conversely, a record is unstable if either a significant fraction of its contents changed recently, or if a series of recent updates affected smaller portions of its contents. An example of a stable record is AF053747.

How useful do you think STABILITY would be in assessing the quality of genomic records? Briefly justify your choice and add comments (if any) in the space provided below.

- ◯ Very useful
- ◯ Somewhat useful
- ◯ Not useful

2. DENSITY

Description: A record is dense if it contains a large amount of information (especially annotations). An example of a dense record is AM270298.

How useful do you think DENSITY would be in assessing the quality of genomic records? Briefly justify your choice and add comments (if any) in the space provided below.

- ◯ Very useful
- ◯ Somewhat useful
- ◯ Not useful

3. FRESHNESS

Description: A record is fresh if it has recently been updated, regardless of the extent of the update (i.e., what fraction of the record's contents changed). An example of a fresh record is NM_001033493.

How useful do you think FRESHNESS would be in assessing the quality of genomic records? Briefly justify your choice and add comments (if any) in the space provided below.

- ◯ Very useful
- ◯ Somewhat useful
- ◯ Not useful

4. LINKAGE

Description: A record that is linked to/from many other records, publications, or resources has high linkage. An example of a record with high linkage is NM_001083617.

How useful do you think LINKAGE would be in assessing the quality of genomic records? Briefly justify your choice and add comments (if any) in the space provided below.

- ◯ Very useful

◯ Somewhat useful
◯ Not useful

5. REDUNDANCY

Description: A record is redundant if other records in the database contain similar information about the sequence represented by the record. An example of a set of redundant records is DX598901, DX598902, DX598903, DX598904, DX598905, and DX598906.

How useful do you think REDUNDANCY would be in assessing the quality of genomic records? Briefly justify your choice and add comments (if any) in the space provided below.

◯ Very useful
◯ Somewhat useful
◯ Not useful

Comparative Study of Automated and Human-based Quality Assessments over

End of Survey / Participant's Information

You have finished completing the survey questionnaire!!
Now we just need to collect your name and mailing address so that we can send you the compensation check.

1. Please enter your name below. **We will address the compensation check to the name you provide here.**
Note: This information will be used for compensation purposes only.

2. Please enter your mailing address below. **We will mail your compensation check to the address you provide here. **
Note: This information will be used for compensation purposes only.

APPENDIX B
SURVEY RESPONSES

Table B-1. Answers to Question 1, Part I of Survey Questionnaire: *Choose from the NCBI's nucleotide or protein databases 12 records whose quality you consider to be "Good" (i.e., above average), and write the accession number of each record in the space provided below (order is not important). Please choose records from different projects and publications*.

Expert's identifier	Expert's answer	
1	1 - AF030859	7 - AF305913
	2 - CO416297	8 - U70480
	3 - Q9SIM9	9 - AJ000759
	4 - S52036	10 - AF394914
	5 - S75487	11 - AB164038
	6 - X77231	12 - Z93766
2	1 - EF051116	7 - AF311734
	2 - AJ812277	8 - DQ144621
	3 - AF024715	9 - AF271234
	4 - AJ489258	10 - AB192966
	5 - AM409201	11 - AY530931
	6 - AY594174	12 - X15656
3	1 - AM700587	7 - AM236040
	2 - AJ784829	8 - XM_364020
	3 - AM421461	9 - AB302211
	4 - AM707022	10 - XM_001465100
	5 - AB302131	11 - XM_001028104
	6 - NM_006926	12 - NM_153171
4	1 - XM_969059.1	7 - XM_001437591.1
	2 - XM_001468499.1	8 - XM_001444703.1
	3 - XM_001468429.1	9 - XM_001458802.1
	4 - XM_001468594.1	10 - BC140538.1
	5 - XM_001468299.1	11 - NM_128106.3
	6 - XM_001466914.1	12 - NM_181070.4
5	1 - AM270121	7 - DN956224
	2 - BC139879	8 - CK348291
	3 - CU234118	9 - CD579131
	4 - BC140635	10 - DN956222
	5 - AAZW01000063	11 - AJ639612
	6 - ES228856	12 - AW670836
6	1 - NP_032304	7 - AAH83149
	2 - NP_033792	8 - NP_000851
	3 - NM_008042	9 - NM_008969
	4 - ABL01512	10 - NP_001034220
	5 - NM_011198	11 - NM_198052
	6 - NM_009660	12 - NM_008517

Table B-1. Continued.

Expert's identifier	Expert's answer	
7	1 - DQ332556	7 - NW_001589540
	2 - DQ200544	8 - AM503065
	3 - EF059394	9 - AF095794
	4 - NM_000059	10 - AF309689
	5 - NM_009844	11 - AADC01011023
	6 - DD322688	12 - EF062220
8	1 - NM_173095	7 - NM_005514
	2 - NM_001042503	8 - NM_006188
	3 - NM_066627	9 - NM_153673
	4 - NM_065114	10 - AY379549
	5 - NM_066249	11 - NM_009324
	6 - AF029303	12 - NM_007305
9	1 - EF555725	7 - AY387313
	2 - EF032775	8 - EF156505
	3 - DQ979194	9 - AB267447
	4 - AY861780	10 - AY207060
	5 - AY519200	11 - AJ270473
	6 - DQ125864	12 - DQ979064
10	1 - AF104363	7 - AF115532
	2 - AF104364	8 - AF373692
	3 - AF104365	9 - AF373693
	4 - AF104366	10 - AF373694
	5 - AF115530	11 - AY273808
	6 - AF115531	12 - DQ288271
11	1 - NM_011891	7 - NM_031504
	2 - NM_023324	8 - EF529813
	3 - XM_747199	9 - EF589044
	4 - NM_014294	10 - NC_009511
	5 - EF139429	11 - NM_003616
	6 - NM_001002964	12 - DQ151655
12	1 - NT_033779	7 - AA392812
	2 - NM_003508	8 - AAF13280
	3 - NT_033777	9 - AAC37178
	4 - NP_649897	10 - AAB59378
	5 - NP_722812	11 - NM_169733
	6 - AAC26105	12 - NP_611323
13	1 - M96972	7 - AF135043
	2 - U15194	8 - U17165
	3 - M93171	9 - K01569
	4 - U80145	10 - U40227
	5 - X13449	11 - M27841
	6 - AY101445	12 - D38129

Table B-1. Continued.

Expert's identifier	Expert's answer	
14	1 - AY666164	7 - NM_001086442
	2 - NM_008356	8 - EF529813
	3 - AB279705	9 - EF418587
	4 - DQ863513	10 - EF077626
	5 - NM_073736	11 - NM_002231
	6 - AL110479	12 - EF615587
15	1 - AY562383	7 - Z49966.
	2 - AB167815.	8 - NM_000879.
	3 - NC_009501.	9 - AY390507.
	4 - EF035538.	10 - NM_204321.
	5 - AY030402.	11 - AY329443.
	6 - EF418587	12 - EF185297.
16	1 - P17561	7 - CAG30452
	2 - Q9UM22	8 - CAA25855
	3 - AF353717	9 - NP_001013128
	4 - Q25472	10 - NM_214575
	5 - P04734	11 - NM_214538
	6 - CAG33109	12 - NM_204920

Table B-2. Answers to Question 2, Part I of Survey Questionnaire: *Choose from the NCBI's nucleotide or protein databases 12 records whose quality you consider to be "Poor" (i.e., below average), and write the accession number of each record in the space provided below (order is not important). Please choose records from different projects and publications.*

Expert's identifier	Expert's answer	
1	1 - AX025477	7 - AY148428
	2 - AY043232	8 - AJ306826
	3 - AY222857	9 - X97853
	4 - Z73946	10 - AF239989
	5 - AJ001681	11 - DQ847149
	6 - AF026267	12 - AF123508
2	1 - AM 69175	7 - DQ845786
	2 - EF 429311	8 - AM498281
	3 - DQ132859	9 - AM176568
	4 - AJ867487	10 - DQ631892
	5 - AB014347	11 - DQ358913
	6 - TYU65089	12 - EF101929
3	1 - EF156409	7 - DQ118014
	2 - CS542023	8 - AB258994
	3 - AC203221	9 - AB231801
	4 - EF486657	10 - AB264051
	5 - EF535228	11 - EF192191
	6 - U61190	12 - AB290840
4	1 - BM864850.2	7 - AJ235805.1
	2 - AL436983.1	8 - NM_004562.1
	3 - CZ169617.1	9 - XM_001033076.2
	4 - NC_009356.1	10 - NM_080562.4
	5 - AF020575.1	11 - CS543085.1
	6 - L12674.1	12 - NM_174874.3
5	1 - X66057	7 - U00096
	2 - NC_001146	8 - AE014073
	3 - NC_001142	9 - U80836
	4 - NC_001136	10 - Z75712
	5 - NM_009458	11 - BA000007
	6 - NC_001147	12 - CR382129
6	1 - NM_007940	7 - O35936
	2 - BAA05120	8 - AAK28625
	3 - AAB25016	9 - AAH24742
	4 - AAB24352	10 - AAQ89268
	5 - AAB22287	11 - NG_001397
	6 - AAA48742	12 - NP_001009963

Table B-2. Continued.

Expert's identifier	Expert's answer	
7	1 - AI975940	7 - XM_001357913
	2 - T14408	8 - DS185054
	3 - AM698098	9 - NC_007645
	4 - BF136800	10 - U66338
	5 - EF058453	11 - AI975600
	6 - BG932368	12 - AA999407
8	1 - M92315	7 - DQ438499
	2 - M96401	8 - DQ009322
	3 - L36968	9 - AB241668
	4 - X67761	10 - DD377352
	5 - X70955	11 - AZ935200
	6 - AQ989463	12 - AB074509
9	1 - AY935163	7 - AY185509
	2 - DQ924857	8 - AF016523
	3 - AY389859	9 - AY098488
	4 - AY096175	10 - EF067828
	5 - DQ219795	11 - EF555516
	6 - DQ520876	12 - DQ170852
10	1 - AF455256	7 - AF104371
	2 - AF455258	8 - AY386254
	3 - AF455259	9 - AY386255
	4 - DQ272153	10 - AY386257
	5 - L24953	11 - AF213323
	6 - L24929	12 - ?????
11	1 - AY813643	7 - CP000713
	2 - NM_001044933	8 - NC_009513
	3 - AM159109	9 - EF589960
	4 - DP000195	10 - AY676118
	5 - NM_001047841	11 - AF310681
	6 - AM183808	12 - NC_003070
12	1 - DX404004	7 - AAQ75417
	2 - AL427955	8 - XP001122518
	3 - CAA58508	9 - NP_035274
	4 - AY402753	10 - ABL63586
	5 - XP001236998	11 - CAJ19243
	6 - ABC66167	12 - CV224730
13	1 - AB162348	7 - DQ455749
	2 - BAD84184	8 - EF408908
	3 - AAM61760	9 - AB128863
	4 - X64147	10 - AY455665
	5 - M99646	11 - AF544975
	6 - Z49180	12 - EF408901
14	1 - AM270351	7 - NT_166530
	2 - NC_004329	8 - AY584752
	3 - NM_008337	9 - NM_001033511 XM_539918
	4 - XM_001031006	10 - EF571582
	5 - DQ060000	11 - NC_009512
	6 - XM_001440354	12 - EF589953

Table B-2. Continued.

Expert's identifier	Expert's answer	
15	1 - Z50116.	7 - EF108313
	2 - AF286313.	8 - EF000001.
	3 - EF202098	9 - EF027094
	4 - DQ388337.	10 - DQ978772.
	5 - EF120880.	11 - EF210470.
	6 - AY152759.	12 - DQ084886.
16	1 - DQ033715	7 - AAZ43086
	2 - XP_418830	8 - CK817283
	3 - AY278559	9 - XP_549467
	4 - AY027861	10 - CV863989
	5 - XP_786460	11 - CR559896
	6 - CAF88218	12 - AAH00686

Table B-3. Answers to Question 1, Part II of Survey Questionnaire: *What criteria do you use when evaluating the quality of a genomic record? If you include several criterions, please rank them in order of importance (1 being the most important, 2 being the second most important, etc). Also, please include a description of each quality factor you provide.*

Expert's identifier	Expert's answer
1	1. Total number of information: tissue where it was isolated, information included in the project title. 2. How clear is the name of the gene. May I straightforward associate a record to another gene? 3. Update of the publications 4. Presence and information about the UTR, often polyA site or signal is not specifically indicated.
2	I work with viruses whose genomic sequence is around three thousand base pairs so it is easy to deal with the genome. I have a general look and define according to the organization of the supplied sequence and how it fits with literature I know off. For example some people define the primers they use and then you find a partial sequence that is outside of these primers as part of the sequence you are dealing with so you know it is a vector and not to be trusted. Other sequences define regions not according to literature and create there own annotation of the genes. Thus I get away from these peoples sequences.
3	1. Detail description of gene or proteins when possible. 2. Detail description of the organism or sample (taxonomy, strain, collection site and date, country, etc). 3. Includes information about publication or several publication that cited the sequence. 4. Details of the programs that were used to predict gene structure.
4	Nucleotide quality (N, W, etc.), partial vs. whole, annotations, reading frame.
5	I was concerned with the revisions. The good records contained 1-2 revisions vs. the bad records contained 20+ revisions.
6	1. The amount of information available of that protein or gene sequence. Says a lot about how reliable the information is and how useful it has been to others. 2. Publication of the whole sequences. Usually incomplete sequences tells us about feasibility of working with that specific protein and/or suggests multiple variants of the same enzyme, suggesting that not everything about that protein is known.
7	Quality Length intron/exon genomic or mRNA.

Table B-3. Continued.

Expert's identifier	Expert's answer
8	1. If there are large numbers of n's/unknown nucleotides in the data (that counts as poor quality) especially if there are sequential stretches of n's or n's at the very end of a sequence, which could easily be cut off when editing, so it indicates perhaps the sequence was not well edited. 2. The amount of information provided about the sequence: 2a. Phylogenetic classification (where expected - e.g. if there are nearly exact genbank species matches yet the sequence is simply classified as "uncultured bacterium," with no attempt at any level of taxonomic classification, then it is poor quality. This is compounded if it is part of a series of hundreds of similar clones that can crowd out the sequence information that has relevant data on it) 2b. Functional information about the gene 1. What it effects/reacts with 2. Related homologs/orthologs 3. Areas of expression in the organism 4. Phenotypic effects 3. Number of publications/if it has been published. The quality of the publication journal can add (ex. Nature), and if it was published by different authors (not overlapping authors in all the groups) that improves the quality as well. 4. Whether it has been recently updated/updated at all, which would indicates the authors are putting in effort to make the information in the entry accurate (this increases the quality)(For example one of the sequences I listed as being poor due to it having outdated information that placed it in an old taxonomic category which conflicted with 98%+ matching sequences listed as the new taxonomic category, making the sequence identification confusing and difficult to cite in a paper). This somewhat overlaps with 2a, but applies more to the area where it mentions if the sequence has been updated.
9	Source of sequence-location or experiment type.
10	Completeness- are there ambiguity characters and is sequence complete in length.
11	Having a complete sequence with minimal "N"'s is the most important thing I look for when evaluating quality.
12	1. Inter- species comparison/conservation 2. Consistency with online database/algorithmic predictions 3. Author concurrence 4. Literature confirmation of experimental data and expansion of/ characterization based on previously published data 5. Personal/local experimental data. thoroughness and depth of data 6. Inter-collegial support.
13	1. Information about the gene structure: if the specifically structures of each framework are clearly showed up that I can choose the exactly sequences I need. 2. For human gene, if their chromosome site is clearly written, that will be nice.
14	1. Easy access to the nucleotide sequence. 2. An explanation of the function and or relationship to other genes or proteins of the gene or protein in question.

Table B-3. Continued.

Expert's identifier	Expert's answer
15	Generally, the more informative the record the better so that it is clear what the submission is and what organism and where it came from. The good records were published, included detailed information about the gene or region submitted and exactly what they are, included information about where the organism was from including host species (for parasitic organisms), included translation information when appropriate.
16	1. Sequence containing undetermined nucleotides (Ns or Xs) 2. Amino acid sequences with frame shifts 3. Chimera sequences (including stretches of amino acids that are not part of the protein) 4. Sequences without a properly assigned name (unnamed protein product), even when their orthologs are known.

Table B-4. Answers to Question 2, Part II of Survey Questionnaire: *How would you rank the overall quality of the following NCBI data sources? Use ranks from 1 (highest quality) to 6 (lowest quality), or N/A[16] (if you have never used the database). You may use the same rank for two or more databases if you consider them to be equivalent in terms of their quality*.

Expert's identifier	dbEST	dbGSS	dbSTS	GenBank	HTC	RefSeq
1	2	3	3	1	3	2
2	2	2		3		1
3	5			5	3	6
4	4	4		1		
5				1		1
6	4	6	5	2	1	3
7	5	4	3	1	2	6
8		1		2		1
9			2	1		1
10				1		1
11	2	2	2	1	2	3
12	2	3		2		3
13	2			1		
14				1		2
15				3		
16	3	4		2	5	1

[16] The *N/A* choice is represented as a blank cell in this table.

Table B-5. Answers to Question 3, Part II of Survey Questionnaire: *When using genomics databases, do you usually work with a well defined set of records on a regular basis or do you work with a different set of records every time? Explain briefly*.

Expert's identifier	Expert's answer
1	I generally switch records because I work on several projects.
2	Well defined set of records.
3	I worked with a defined group of organisms and one specific gene but not a specific group of records. I worked with environmental samples, all of them are from the same kingdom and same gene. I compared my data with many different papers or records in GenBank.
4	Different.
5	I work with a different set every time. I do this because it I want to set up in silico experiments with random samples like I do with microarray samples.
6	I usually work with a defined set of records related to my research.
7	Typically I work with a defined set of records based on the project. However, I do work with different sets of records for each project.
8	I usually work with a well-defined set of records, such as cyanobacteria/bacteria and c. elegans, with occasional branches out to look for homologs of interesting genes, usually in more complex organisms (mice, humans). Basically the set will focus on the organism/taxa of interest, and only change if I change major projects (every few years).
9	I'm probing diversity of my sequences so generally different records.
10	Different databases and genes.
11	I work with an organism which doesn't have a fully sequenced genome so I usually work without a well defined set of records.
12	Try to compare and contrast multiple databases/literatures for consistency.
13	I usually work with a well defined set of records because I check the sequences of other species of my gene from the published paper. The work from the well-known paper are always believable and useful.
14	Usually well defined set of records.
15	For most work, I have multiple sets of defined records that I work with on a daily basis. These are sequences that I use as reference sequences.
16	I work with a well defined set of records. But a new sequence for my favorite protein is available from different organism; I must analyze it to determine its quality and if it is truly a new member of the protein family.

Table B-6. Answers to Question 4, Part II of Survey Questionnaire: *Do you know of any genomics database that provides quality scores for its data? If so, please mention it here.*

Expert's identifier	Expert's answer
1	SGN provides quality scores for the markers.
2	No.
3	No.
4	No.
5	Most of the time I deal with p-values.
6	No.
7	I don't think any of the databases I use on a regular basis provide quality scores. If they do I'm not sure I understand them.
8	None that I know of.
9	Greengenes collects a subset of "good" 16s environmental sequences from Genbank and RDP.
10	No.
11	I am unaware of any databases which do provide quality scores.
12	Some databases, e.g., yeast databases provide p-values for comparison accuracy.
13	NCBI is the mainly database I work with, I believe in it most of time.
14	No I don't.
15	No.
16	No, I not aware of that kind of database.

Table B-7. Answers to Question 1, Part III of Survey Questionnaire: *How useful do you think STABILITY would be in assessing the quality of genomic records?*[17] *Briefly justify your choice and add comments (if any) in the space provided below.*

Expert's identifier	Expert's rating	Expert's comments
1	S	A database that is quickly updated is a lot better than one in which records are fossilised.
2	N	The sequence is much more informative and should be stable but the annotation may change as more knowledge becomes available and it is important to have a standard criteria for the annotation with the availability of more sequences and knowledge. Thus in order to keep a sequence informative the gathering of knowledge and references in one place is very important.
3	S	It is impossible to have only stable records because people obtain new information about their data (which is good) but a record with changes in primary information such as organism, collection sites, strains is not useful.
4	S	Genome sequencing tends to change the database and retire genes; it would be nice if all sequences were stable, but lets face it some areas are just hard to sequence thru.
5	V	As long as a curator does not take away any old information from records I consider them to be quite stable.
6	V	It suggests the information is reliable and that it has been corroborated by different investigators.
7	V	It is important to know when information has changed as it might affect the results of any project you might be using the record for.
8	N	Although I think it is important to know whether or not a record has been updated/changed, if it has changed, that would be a reflection on the quality of the old record, not necessarily the new one. It would be good to have for evaluating older papers/references that might have used the old data, but would not necessarily be of use to the new sequence. Also, how stable it was would to some degree depend more on how much new research was being done on that sequence/organism rather than the relative quality of the sequence itself (e.g. a sequence could be very low quality, but if no one else researches the organism for 10 years, it may be very stable).
9	V	Good records are complete and don't need continual updating.
10	V	Stability would be very useful, if things change often, especially sequence, then the results of projects could change.
11	V	Working with unstable accession ID's can be confusing and time consuming.

[17] The usefulness was assessed in a 3-point scale: very useful (V), somewhat useful (S), and not useful (N).

Table B-7. Continued.

12	S	The fact that the data has not been contested/redefined in several years suggests that it is accurate; however, if no significant progress or analysis has been carried out for great time periods, i.e. within 2-3 years or more, it is indicative of lack of analysis rather than stability of data.
13	S	It depends. As long as the information is right at that time, updating is ok.
14	V	If a record stays unchanged it shows the quality of the sequencing.
15	N	If a record does not change it often means that there is no work being done and thus no errors ARE found. In this case, the "stable" record could be worse.
16	S	The usefulness of stability for sequence records will depend on the specific records studied. I'm glad that several records that I knew had some problems were corrected in later updates of GenBank. But others have stayed unchanged for almost 7 years, even though I have evidence that they are incorrect.

Table B-8. Answers to Question 2, Part III of Survey Questionnaire: *How useful do you think DENSITY would be in assessing the quality of genomic records?* [18] *Briefly justify your choice and add comments (if any) in the space provided below.*

Expert's identifier	Expert's rating	Expert's comments
1	S	That record is too chaotic. It may be good for a first screen but the specific gene sections are more important.
2	V	The more the sequence knowledge is available for the user the more it is to look at and make use of it and this provides like a comfortable or relaxing feeling that you do not have to look into the details of the sequence to determine what you need especially in the case of chromosome sequences.
3	V	Detail information (lots of annotations) is specially important in cases when the paper has not been published and it is very useful for comparisons with other datasets (e.g. studies of environmental samples).
4	N	Though it would be nice to have and extra link to get to 'all' the information, extremely dense records seem unnecessary.
5	V	I would consider this to be very useful in assessing the quality of genomic records. I believe that any type of ancillary data, especially extra literature, to be very useful as opposed to very little information.
6	V	It suggests the sequence is well known and characterized, which makes experiment design easier. It gives you a clearer idea as to whether the sequence you are dealing with is unique or could have variants.
7	N	I think these dense records would be better if you cold link to the information from one page, rather than scrolling for miles to find the information you need.
8	V	The more publications and information on the gene there is, almost certainly the more useful and higher quality the sequence, since it has been presumably checked over multiple times by different researchers and reviewed by various journals. It is also of greater use to a researcher attempting to obtain information about the gene/its location/its homologs, etc. It might be nice if there could be some redundancy of author check as well, since if there were 5 publications all with one author in it, I would consider that lower quality than 5 different authors publishing the same sequence.
9	S	Some dense records just have useless excess info.
10	V	This would be nice because it shows papers and projects that have used the sequence.
11	S	More information can be valuable or overwhelming.
12	S	Multiple data sets can be helpful in identifying multiple isoforms/transcripts of individual genes. Conversely, they may indicate potential for more confined, specific sequence requirement for true expression, or for tissue specific expression, etc. Or the data could represent conflicting reports/views/data sets from competing laboratories.

[18] The usefulness was assessed in a 3-point scale: very useful (V), somewhat useful (S), and not useful (N).

Table B-8. Continued.

Expert's identifier	Expert's rating	Expert's comments
13	S	Information is surely useful, but sometimes too much information in one record will make user lost.
14	S	Density can be good to obtain information on the record and how it was obtained, but it can get cumbersome to understand.
15	V	Dense usually means that a lot of information is included, and much of it is likely useful.
16	V	Density is very useful because most bibliographic and sequence information is linked to the record. In addition, you can know in a glance the function, and special features of the gene/protein.

Table B-9. Answers to Question 3, Part III of Survey Questionnaire: *How useful do you think FRESHNESS would be in assessing the quality of genomic records?* [19] *Briefly justify your choice and add comments (if any) in the space provided below.*

Expert's identifier	Expert's rating	Expert's comments
1	V	It is necessary when there are similar sequences: are they sequencing errors? SNPs or different genes or whatelse? The updates as presented in the acc number presented are perfect.
2	N	It is not useful because the updating of all sequences in the gene bank is not practical so it depends on the field you are working in and how much this update is fresh. But in general this is not the bases.
3	N	It is good to have the last version of a record but it is not essential to evaluate the quality of the record.
4	N	Seems unnecessary.
5	V	I use freshness as additions to old information mostly for updates.
6	V	Things always change and people make mistakes. Updates are always good and very informative.
7	V	It is important to have all the information available.
8	S	I think it would be an important factor to know if a sequence had been updated recently, because, as stated before, it would indicate that effort was being put into to keeping the record up-to-date and accurate. However, this measure could put initially very high-quality records at a disadvantage.
9	S	This type of updating isn't common in the records that I generally use so I'm not sure how useful this would be.
10	V	Knowing when things have been updated is critical.
11	S	Intuitively it seems that a record with more freshness would help in determining something's quality.
12	S	Previously characterized data has, on occasion, been shown to be misinterpreted, in which case maintaining a fresh database can be extremely valuable to researchers who might be struggling for answers to mysteriously unanswerable questions. However, data that was posted in the past, say five years ago or more, could be either because data analysis has experimentally stood the test of time, or, by contrast, it could indicate that the molecule has remained unstudied since initial publication, that no experiments have been conducted to prove, disprove, or further characterize the molecule.
13	V	The information is always updated, so to show the update information is very important to user.
14	V	If a record has changed it should be stated.
15	S	I think this could be useful because it indicates that work is ongoing and theoretically the information is increasing or getting better.
16	V	Freshness is very useful because genes not previously linked protein domains or recent literature can be identified.

[19] The usefulness was assessed in a 3-point scale: very useful (V), somewhat useful (S), and not useful (N).

Table B-10. Answers to Question 4, Part III of Survey Questionnaire: *How useful do you think LINKAGE would be in assessing the quality of genomic records?* [20] *Briefly justify your choice and add comments (if any) in the space provided below.*

Expert's identifier	Expert's rating	Expert's comments
1	V	It helps a lot. I think they are complementary.
2	S	What it is linked this.
3	V	This is very useful because you can obtain a lot of information about that record.
4	N	I would like to see a separate link to access that stuff.
5	V	I always teach my students to include everything. New information should always be used to qualify them in papers, posters, and reports.
6	V	Same as question 2. [It suggests the sequence is well known and characterized, which makes experiment design easier. It gives you a clearer idea as to whether the sequence you are dealing with is unique or could have variants].
7	V	This demonstrates how many people have done work on this sequence, which in turn relays the importance and accuracy of the sequences.
8	V	I think a record with high linkage is almost always of greater quality and usefulness than one of low linkage, again because it has been evaluated multiple times and presumably has more information attached to it to assist a researcher with learning about it.
9	V	Well linked records indicate through prep of the sequences/metadata.
10	V	This would be nice, but have not used it.
11	V	Having quick links readily accessible enhance quality assessment.
12	V	The more publications about a gene/protein, the more research has been conducted, and the more is known about its characteristics, expression patterns, and functions. More extensive knowledge could be of tremendous benefit to medicine, even if more recent data negate initial conclusions. It may also be useful in that a gene/protein that receives more attention may be indicative of necessity to obtain detailed answers swiftly and precisely.
13	V	This is the most part I like. I really would like to check the publications and resources conveniently. That helps my work a lot.
14	V	Very useful to know what other publications are linked to one record.
15	S	This may be useful because it indicates the authors have examined many similar items rather than just one.
16	V	Linkage is of fundamental importance, because most of what is know about a record can be assessed in a glance.

[20] The usefulness was assessed in a 3-point scale: very useful (V), somewhat useful (S), and not useful (N).

Table B-11. Answers to Question 5, Part III of Survey Questionnaire: *How useful do you think REDUNDANCY would be in assessing the quality of genomic records?* [21] *Briefly justify your choice and add comments (if any) in the space provided below.*

Expert's identifier	Expert's rating	Expert's comments
1	N	If they are exactly the same they are useless and troublesome.
2	S	The sequence if there and is redundant this means that the sequencing process is to be trusted. However this is extremely important in some cases for the study of genetic diversity and population genetics who look at point mutations but from a practical point of view for genes and working with them it is somehow important to be sure from several sequences especially in case of pathogens or important conserved domains that they are stable and the sequence is repetitive.
3	N	If it is exactly the same sequence, redundancy makes searches really complex. It would be better to have links instead getting pages and pages of the same hits.
4	N	I would like to see it all combined, and I would appreciate it if all the sequences that link together for one gene were submitted in one file.
5	N	N/A
6	V	It helps corroborates the published data. Although it could be misleading.
7	N	I think redundant records should be combined until information is provided that the record can stand alone.
8	S	As with any scientific data, the greater the ability to replicate, the higher the quality of the data. However, I again would be concerned if the set of records was all from one author or not - I've found many sets of hundreds of similar sequences all from one project which I would not necessarily rank as highly in adding to the quality of a sequence (though it would still be better than having no replicates).
9	N	Not useful.
10	S	Don't know.
11	N	Redundancy seems redundant.
12	V	Redundancy could potentially indicate gene duplications that resulted in multi-tissue expression, with either redundant, novel, or pleiotropic functions.
13	S	It depends. Meaningless repeat is just a waste and sometimes just disturb user's eyes. But sometimes similar records exist and maybe bring user information.
14	S	Usually good to know how consistent a sequence is.
15	S	This may be useful to allow you to compare the redundant records, but I do not think it indicates much about quality itself.

[21] The usefulness was assessed in a 3-point scale: very useful (V), somewhat useful (S), and not useful (N).

Table B-11. Continued.

Expert's identifier	Expert's rating	Expert's comments
16	S	Redundancy is useful if it comes from cDNA sequencing from several tissues, because you can determine if there are tissue specific variants, or if they represent several readings of a gene, and you can construct a contig from them. Redundancy is not useful if the tissue used was the same.

APPENDIX C
ORIGINAL INSDSEQ XML SCHEMA

We append here the original INSDSeq XML Schema (Version 1.4, 19 September 2005) automatically generated by the DATATOOL utility (version 1.8.1, 18 January 2007).

```
<?xml version="1.0" ?>
<!-- ============================================
    ::DATATOOL:: Generated from "insdseq.asn"
    ::DATATOOL:: by application DATATOOL version 1.8.1
    ::DATATOOL:: on 01/18/2007 23:07:18
    ============================================ -->

<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="http://www.ncbi.nlm.nih.gov"
  targetNamespace="http://www.ncbi.nlm.nih.gov"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified">

<!-- ============================================ -->
<!-- This section is mapped from module "INSD-INSDSeq"
============================================== -->
<!--
$Revision: 1.6 $
*****************************************************************************

 ASN.1 and XML for the components of a GenBank/EMBL/DDBJ sequence record
 The International Nucleotide Sequence Database (INSD) collaboration
 Version 1.4, 19 September 2005

*****************************************************************************
-->


<!--
  INSDSeq provides the elements of a sequence as presented in the
    GenBank/EMBL/DDBJ-style flatfile formats, with a small amount of
    additional structure.
    Although this single perspective of the three flatfile formats
    provides a useful simplification, it hides to some extent the
    details of the actual data underlying those formats. Nevertheless,
    the XML version of INSD-Seq is being provided with
    the hopes that it will prove useful to those who bulk-process
```

sequence data at the flatfile-format level of detail. Further documentation regarding the content and conventions of those formats can be found at:

URLs for the DDBJ, EMBL, and GenBank Feature Table Document:
http://www.ddbj.nig.ac.jp/FT/full_index.html
http://www.ebi.ac.uk/embl/Documentation/FT_definitions/feature_table.html
http://www.ncbi.nlm.nih.gov/projects/collab/FT/index.html

URLs for DDBJ, EMBL, and GenBank Release Notes :
ftp://ftp.ddbj.nig.ac.jp/database/ddbj/ddbjrel.txt
http://www.ebi.ac.uk/embl/Documentation/Release_notes/current/relnotes.html
ftp://ftp.ncbi.nih.gov/genbank/gbrel.txt

Because INSDSeq is a compromise, a number of pragmatic decisions have been made:

In pursuit of simplicity and familiarity a number of fields do not have full substructure defined here where there is already a standard flatfile format string. For example:

Dates: DD-MON-YYYY (eg 10-JUN-2003)

Author: LastName, Initials (eg Smith, J.N.)
or Lastname Initials (eg Smith J.N.)

Journal: JournalName Volume (issue), page-range (year)
or JournalName Volume(issue):page-range(year)
eg Appl. Environ. Microbiol. 61 (4), 1646-1648 (1995)
Appl. Environ. Microbiol. 61(4):1646-1648(1995).

FeatureLocations are representated as in the flatfile feature table, but FeatureIntervals may also be provided as a convenience

FeatureQualifiers are represented as in the flatfile feature table.

Primary has a string that represents a table to construct a third party (TPA) sequence.

other-seqids can have strings with the "vertical bar format" sequence identifiers used in BLAST for example, when they are non-INSD types.

Currently in flatfile format you only see Accession numbers, but there are others, like patents, submitter clone names, etc which will appear here

There are also a number of elements that could have been more exactly specified, but in the interest of simplicity have been simply left as optional. For example:

All publicly accessible sequence records in INSDSeq format will include accession and accession.version. However, these elements are optional in optional in INSDSeq so that this format can also be used for non-public sequence data, prior to the assignment of accessions and version numbers. In such cases, records will have only "other-seqids".

sequences will normally all have "sequence" filled in. But contig records will have a "join" statement in the "contig" slot, and no "sequence". We also may consider a retrieval option with no sequence of any kind and no feature table to quickly check minimal values.

Four (optional) elements are specific to records represented via the EMBL sequence database: INSDSeq_update-release, INSDSeq_create-release, INSDSeq_entry-version, and INSDSeq_database-reference.

One (optional) element is specific to records originating at the GenBank and DDBJ sequence databases: INSDSeq_segment.

****
-->

```
<xs:element name="INSDSeq">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="INSDSeq_locus" type="xs:string"/>
   <xs:element name="INSDSeq_length" type="xs:integer"/>
   <xs:element name="INSDSeq_strandedness" type="xs:string" minOccurs="0"/>
   <xs:element name="INSDSeq_moltype" type="xs:string"/>
   <xs:element name="INSDSeq_topology" type="xs:string" minOccurs="0"/>
   <xs:element name="INSDSeq_division" type="xs:string"/>
   <xs:element name="INSDSeq_update-date" type="xs:string"/>
   <xs:element name="INSDSeq_create-date" type="xs:string" minOccurs="0"/>
   <xs:element name="INSDSeq_update-release" type="xs:string" minOccurs="0"/>
   <xs:element name="INSDSeq_create-release" type="xs:string" minOccurs="0"/>
   <xs:element name="INSDSeq_definition" type="xs:string"/>
   <xs:element name="INSDSeq_primary-accession" type="xs:string" minOccurs="0"/>
   <xs:element name="INSDSeq_entry-version" type="xs:string" minOccurs="0"/>
   <xs:element name="INSDSeq_accession-version" type="xs:string" minOccurs="0"/>
   <xs:element name="INSDSeq_other-seqids" minOccurs="0">
    <xs:complexType>
     <xs:sequence minOccurs="0" maxOccurs="unbounded">
      <xs:element ref="INSDSeqid"/>
     </xs:sequence>
```

```
      </xs:complexType>
    </xs:element>
    <xs:element name="INSDSeq_secondary-accessions" minOccurs="0">
      <xs:complexType>
        <xs:sequence minOccurs="0" maxOccurs="unbounded">
          <xs:element ref="INSDSecondary-accn"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="INSDSeq_project" type="xs:string" minOccurs="0"/>
    <xs:element name="INSDSeq_keywords" minOccurs="0">
      <xs:complexType>
        <xs:sequence minOccurs="0" maxOccurs="unbounded">
          <xs:element ref="INSDKeyword"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="INSDSeq_segment" type="xs:string" minOccurs="0"/>
    <xs:element name="INSDSeq_source" type="xs:string" minOccurs="0"/>
    <xs:element name="INSDSeq_organism" type="xs:string" minOccurs="0"/>
    <xs:element name="INSDSeq_taxonomy" type="xs:string" minOccurs="0"/>
    <xs:element name="INSDSeq_references" minOccurs="0">
      <xs:complexType>
        <xs:sequence minOccurs="0" maxOccurs="unbounded">
          <xs:element ref="INSDReference"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="INSDSeq_comment" type="xs:string" minOccurs="0"/>
    <xs:element name="INSDSeq_primary" type="xs:string" minOccurs="0"/>
    <xs:element name="INSDSeq_source-db" type="xs:string" minOccurs="0"/>
    <xs:element name="INSDSeq_database-reference" type="xs:string" minOccurs="0"/>
    <xs:element name="INSDSeq_feature-table" minOccurs="0">
      <xs:complexType>
        <xs:sequence minOccurs="0" maxOccurs="unbounded">
          <xs:element ref="INSDFeature"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
<!-- Optional for other dump forms -->
    <xs:element name="INSDSeq_sequence" type="xs:string" minOccurs="0"/>
    <xs:element name="INSDSeq_contig" type="xs:string" minOccurs="0"/>
   </xs:sequence>
  </xs:complexType>
</xs:element>
```

```
<xs:element name="INSDSeqid" type="xs:string"/>

<xs:element name="INSDSecondary-accn" type="xs:string"/>

<xs:element name="INSDKeyword" type="xs:string"/>

<!--
 INSDReference_position contains a string value indicating the
 basepair span(s) to which a reference applies. The allowable
 formats are:

   X..Y  : Where X and Y are integers separated by two periods,
           X >= 1 , Y <= sequence length, and X <= Y

           Multiple basepair spans can exist, separated by a
           semi-colon and a space. For example : 10..20; 100..500

   sites : The string literal 'sites', indicating that a reference
           provides sequence annotation information, but the specific
           basepair spans are either not captured, or were too numerous
           to record.

           The 'sites' literal string is singly occuring, and
            cannot be used in conjunction with any X..Y basepair spans.

   References that lack an INSDReference_position element apply
   to the entire sequence.
-->
<xs:element name="INSDReference">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="INSDReference_reference" type="xs:string"/>
      <xs:element name="INSDReference_position" type="xs:string" minOccurs="0"/>
      <xs:element name="INSDReference_authors" minOccurs="0">
        <xs:complexType>
          <xs:sequence minOccurs="0" maxOccurs="unbounded">
            <xs:element ref="INSDAuthor"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="INSDReference_consortium" type="xs:string" minOccurs="0"/>
      <xs:element name="INSDReference_title" type="xs:string" minOccurs="0"/>
      <xs:element name="INSDReference_journal" type="xs:string"/>
      <xs:element name="INSDReference_xref" minOccurs="0">
        <xs:complexType>
          <xs:sequence minOccurs="0" maxOccurs="unbounded">
```

```
          <xs:element ref="INSDXref"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="INSDReference_pubmed" type="xs:integer" minOccurs="0"/>
    <xs:element name="INSDReference_remark" type="xs:string" minOccurs="0"/>
   </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="INSDAuthor" type="xs:string"/>

<!--
 INSDXref provides a method for referring to records in
 other databases. INSDXref_dbname is a string value that
 provides the name of the database, and INSDXref_dbname
 is a string value that provides the record's identifier
 in that database.
-->
<xs:element name="INSDXref">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="INSDXref_dbname" type="xs:string"/>
      <xs:element name="INSDXref_id" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<!--
 INSDFeature_operator contains a string value describing
 the relationship among a set of INSDInterval within
 INSDFeature_intervals. The allowable formats are:

   join :  The string literal 'join' indicates that the
           INSDInterval intervals are biologically joined
           together into a contiguous molecule.

   order : The string literal 'order' indicates that the
           INSDInterval intervals are in the presented
           order, but they are not necessarily contiguous.

   Either 'join' or 'order' is required if INSDFeature_intervals
   is comprised of more than one INSDInterval .
-->
<xs:element name="INSDFeature">
  <xs:complexType>
```

```
    <xs:sequence>
      <xs:element name="INSDFeature_key" type="xs:string"/>
      <xs:element name="INSDFeature_location" type="xs:string"/>
      <xs:element name="INSDFeature_intervals" minOccurs="0">
        <xs:complexType>
          <xs:sequence minOccurs="0" maxOccurs="unbounded">
            <xs:element ref="INSDInterval"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="INSDFeature_operator" type="xs:string" minOccurs="0"/>
      <xs:element name="INSDFeature_partial5" minOccurs="0">
        <xs:complexType>
          <xs:attribute name="value" use="required">
            <xs:simpleType>
              <xs:restriction base="xs:string">
                <xs:enumeration value="true"/>
                <xs:enumeration value="false"/>
              </xs:restriction>
            </xs:simpleType>
          </xs:attribute>
        </xs:complexType>
      </xs:element>
      <xs:element name="INSDFeature_partial3" minOccurs="0">
        <xs:complexType>
          <xs:attribute name="value" use="required">
            <xs:simpleType>
              <xs:restriction base="xs:string">
                <xs:enumeration value="true"/>
                <xs:enumeration value="false"/>
              </xs:restriction>
            </xs:simpleType>
          </xs:attribute>
        </xs:complexType>
      </xs:element>
      <xs:element name="INSDFeature_quals" minOccurs="0">
        <xs:complexType>
          <xs:sequence minOccurs="0" maxOccurs="unbounded">
            <xs:element ref="INSDQualifier"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

```
<!--
 INSDInterval_iscomp is a boolean indicating whether
 an INSDInterval_from / INSDInterval_to location
 represents a location on the complement strand.
 When INSDInterval_iscomp is TRUE, it essentially
 confirms that a 'from' value which is greater than
 a 'to' value is intentional, because the location
 is on the opposite strand of the presented sequence.
 INSDInterval_interbp is a boolean indicating whether
 a feature (such as a restriction site) is located
 between two adjacent basepairs. When INSDInterval_iscomp
 is TRUE, the 'from' and 'to' values must differ by
 exactly one base.
-->
<xs:element name="INSDInterval">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="INSDInterval_from" type="xs:integer" minOccurs="0"/>
      <xs:element name="INSDInterval_to" type="xs:integer" minOccurs="0"/>
      <xs:element name="INSDInterval_point" type="xs:integer" minOccurs="0"/>
      <xs:element name="INSDInterval_iscomp" minOccurs="0">
        <xs:complexType>
          <xs:attribute name="value" use="required">
            <xs:simpleType>
              <xs:restriction base="xs:string">
                <xs:enumeration value="true"/>
                <xs:enumeration value="false"/>
              </xs:restriction>
            </xs:simpleType>
          </xs:attribute>
        </xs:complexType>
      </xs:element>
      <xs:element name="INSDInterval_interbp" minOccurs="0">
        <xs:complexType>
          <xs:attribute name="value" use="required">
            <xs:simpleType>
              <xs:restriction base="xs:string">
                <xs:enumeration value="true"/>
                <xs:enumeration value="false"/>
              </xs:restriction>
            </xs:simpleType>
          </xs:attribute>
        </xs:complexType>
      </xs:element>
      <xs:element name="INSDInterval_accession" type="xs:string"/>
    </xs:sequence>
```

```
  </xs:complexType>
</xs:element>

<xs:element name="INSDQualifier">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="INSDQualifier_name" type="xs:string"/>
      <xs:element name="INSDQualifier_value" type="xs:string" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="INSDSet">
  <xs:complexType>
    <xs:sequence minOccurs="0" maxOccurs="unbounded">
      <xs:element ref="INSDSeq"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

</xs:schema>
```

APPENDIX D
INSDSEQ_QM XML SCHEMA

We append here the XML Schema used by the Local Cache in the Quality Management Architecture, named INSDSeq_QM.xsd. This is a modified version of the INSDSeq schema shown in Appendix C. We have removed all the lengthy comments that appeared in the original version of Appendix C for the sake of brevity.

```
<?xml version="1.0" ?>
<!-- ============================================
    ::DATATOOL:: Generated from "insdseq.asn"
    ::DATATOOL:: by application DATATOOL version 1.8.1
    ::DATATOOL:: on 01/18/2007 23:07:18
    ============================================ -->

<!-- ============================================
    Modified on 06/18/2007 by Alexandra Martinez
    Changes were made to adapt this schema for
    the BIODQ Project's Quality Metadata Engine
    ============================================ -->

<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:xdb="http://xmlns.oracle.com/xdb"
  xmlns="http://biodq.cise.ufl.edu"
  targetNamespace="http://biodq.cise.ufl.edu"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified" >

<!--
$Revision: 1.6 $
*****************************************************************************

 ASN.1 and XML for the components of a GenBank/EMBL/DDBJ sequence record
 The International Nucleotide Sequence Database (INSD) collaboration
 Version 1.4, 19 September 2005

*****************************************************************************
-->

<!-- ========================================
     COMPLEX TYPES ADDED for BIODQ Project
```

```
    ========================================== -->

<xs:complexType name="stringWithId" xdb:SQLType="STRING_T">
   <xs:simpleContent>
      <xs:extension base="xs:string">
         <xs:attribute name="id" type="xs:ID" use="required"/>
      </xs:extension>
   </xs:simpleContent>
</xs:complexType>

<xs:complexType name="integerWithId" xdb:SQLType="INTEGER_T">
   <xs:simpleContent>
      <xs:extension base="xs:integer">
         <xs:attribute name="id" type="xs:ID" use="required"/>
      </xs:extension>
   </xs:simpleContent>
</xs:complexType>


<!-- Modified Schema elements -->

<!-- Root element: represents a record -->
<xs:element name="INSDSeq" xdb:defaultTable="INSDSEQ">
  <xs:complexType xdb:SQLType="INSDSEQ_T">
   <xs:sequence>
     <xs:element name="INSDSeq_locus" type="stringWithId"/>
     <xs:element name="INSDSeq_length" type="integerWithId"/>
     <xs:element name="INSDSeq_strandedness" type="stringWithId" minOccurs="0"/>
     <xs:element name="INSDSeq_moltype" type="stringWithId"/>
     <xs:element name="INSDSeq_topology" type="stringWithId" minOccurs="0"/>
     <xs:element name="INSDSeq_division" type="stringWithId"/>
     <xs:element name="INSDSeq_update-date" type="stringWithId"/>
     <xs:element name="INSDSeq_create-date" type="stringWithId" minOccurs="0"/>
     <xs:element name="INSDSeq_update-release" type="stringWithId" minOccurs="0"/>
     <xs:element name="INSDSeq_create-release" type="stringWithId" minOccurs="0"/>
     <xs:element name="INSDSeq_definition" type="stringWithId"/>
     <xs:element name="INSDSeq_primary-accession" type="stringWithId" minOccurs="0"/>
     <xs:element name="INSDSeq_entry-version" type="stringWithId" minOccurs="0"/>
     <xs:element name="INSDSeq_accession-version" type="stringWithId" minOccurs="0"/>
     <xs:element name="INSDSeq_other-seqids" minOccurs="0">
       <xs:complexType xdb:SQLType="OTHER-SEQIDS_T">
         <xs:sequence minOccurs="0" maxOccurs="unbounded">
           <xs:element ref="INSDSeqid"/>
         </xs:sequence>
         <xs:attribute name="id" type="xs:ID" use="required"/>
       </xs:complexType>
```

```
    </xs:element>
    <xs:element name="INSDSeq_secondary-accessions" minOccurs="0">
     <xs:complexType xdb:SQLType="SECONDARY-ACCS_T">
      <xs:sequence minOccurs="0" maxOccurs="unbounded">
       <xs:element ref="INSDSecondary-accn"/>
      </xs:sequence>
      <xs:attribute name="id" type="xs:ID" use="required"/>
     </xs:complexType>
    </xs:element>
    <xs:element name="INSDSeq_project" type="stringWithId" minOccurs="0"/>
    <xs:element name="INSDSeq_keywords" minOccurs="0">
     <xs:complexType xdb:SQLType="KEYWORDS_T">
      <xs:sequence minOccurs="0" maxOccurs="unbounded">
       <xs:element ref="INSDKeyword"/>
      </xs:sequence>
      <xs:attribute name="id" type="xs:ID" use="required"/>
     </xs:complexType>
    </xs:element>
    <xs:element name="INSDSeq_segment" type="stringWithId" minOccurs="0"/>
    <xs:element name="INSDSeq_source" type="stringWithId" minOccurs="0"/>
    <xs:element name="INSDSeq_organism" type="stringWithId" minOccurs="0"/>
    <xs:element name="INSDSeq_taxonomy" type="stringWithId" minOccurs="0"/>
    <xs:element name="INSDSeq_references" minOccurs="0">
     <xs:complexType xdb:SQLType="REFERENCES_T">
      <xs:sequence minOccurs="0" maxOccurs="unbounded">
       <xs:element ref="INSDReference"/>
      </xs:sequence>
      <xs:attribute name="id" type="xs:ID" use="required"/>
     </xs:complexType>
    </xs:element>
    <xs:element name="INSDSeq_comment" type="stringWithId" minOccurs="0"/>
    <xs:element name="INSDSeq_primary" type="stringWithId" minOccurs="0"/>
    <xs:element name="INSDSeq_source-db" type="stringWithId" minOccurs="0"/>
    <xs:element name="INSDSeq_database-reference" type="stringWithId" minOccurs="0"/>
    <xs:element name="INSDSeq_feature-table" minOccurs="0">
     <xs:complexType xdb:SQLType="FEATURE_TABLE_T">
      <xs:sequence minOccurs="0" maxOccurs="unbounded">
       <xs:element ref="INSDFeature"/>
      </xs:sequence>
      <xs:attribute name="id" type="xs:ID" use="required"/>
     </xs:complexType>
    </xs:element>
<!-- Optional for other dump forms -->
    <xs:element name="INSDSeq_sequence" type="stringWithId" minOccurs="0"/>
    <xs:element name="INSDSeq_contig" type="stringWithId" minOccurs="0"/>
   </xs:sequence>
```

```
   <xs:attribute name="id" type="xs:ID" use="required"/>
  </xs:complexType>
</xs:element>

<xs:element name="INSDSeqid" type="stringWithId" xdb:defaultTable=""/>

<xs:element name="INSDSecondary-accn" type="stringWithId" xdb:defaultTable=""/>

<xs:element name="INSDKeyword" type="stringWithId" xdb:defaultTable=""/>

<xs:element name="INSDReference">
  <xs:complexType xdb:SQLType="REFERENCE_T">
    <xs:sequence>
      <xs:element name="INSDReference_reference" type="stringWithId"/>
      <xs:element name="INSDReference_position" type="stringWithId" minOccurs="0"/>
      <xs:element name="INSDReference_authors" minOccurs="0">
        <xs:complexType xdb:SQLType="AUTHORS_T">
          <xs:sequence minOccurs="0" maxOccurs="unbounded">
            <xs:element ref="INSDAuthor"/>
          </xs:sequence>
          <xs:attribute name="id" type="xs:ID" use="required"/>
        </xs:complexType>
      </xs:element>
      <xs:element name="INSDReference_consortium" type="stringWithId" minOccurs="0"/>
      <xs:element name="INSDReference_title" type="stringWithId" minOccurs="0"/>
      <xs:element name="INSDReference_journal" type="stringWithId"/>
      <xs:element name="INSDReference_xref" minOccurs="0">
        <xs:complexType xdb:SQLType="XREFS_T">
          <xs:sequence minOccurs="0" maxOccurs="unbounded">
            <xs:element ref="INSDXref"/>
          </xs:sequence>
          <xs:attribute name="id" type="xs:ID" use="required"/>
        </xs:complexType>
      </xs:element>
      <xs:element name="INSDReference_pubmed" type="integerWithId" minOccurs="0"/>
      <xs:element name="INSDReference_remark" type="stringWithId" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="id" type="xs:ID" use="required"/>
  </xs:complexType>
</xs:element>

<xs:element name="INSDAuthor" type="stringWithId"/>

<xs:element name="INSDXref" xdb:defaultTable="">
  <xs:complexType xdb:SQLType="XREF_T">
    <xs:sequence>
```

```
      <xs:element name="INSDXref_dbname" type="stringWithId"/>
      <xs:element name="INSDXref_id" type="stringWithId"/>
    </xs:sequence>
    <xs:attribute name="id" type="xs:ID" use="required"/>
  </xs:complexType>
</xs:element>

<xs:element name="INSDFeature">
  <xs:complexType xdb:SQLType="FEATURE_T">
    <xs:sequence>
      <xs:element name="INSDFeature_key" type="stringWithId"/>
      <xs:element name="INSDFeature_location" type="stringWithId"/>
      <xs:element name="INSDFeature_intervals" minOccurs="0">
        <xs:complexType xdb:SQLType="INTERVALS_T">
          <xs:sequence minOccurs="0" maxOccurs="unbounded">
            <xs:element ref="INSDInterval"/>
          </xs:sequence>
          <xs:attribute name="id" type="xs:ID" use="required"/>
        </xs:complexType>
      </xs:element>
      <xs:element name="INSDFeature_operator" type="stringWithId" minOccurs="0"/>
      <xs:element name="INSDFeature_partial5" minOccurs="0">
        <xs:complexType xdb:SQLType="PARTIAL5_T">
          <xs:attribute name="value" use="required">
            <xs:simpleType>
              <xs:restriction base="xs:string">
                <xs:enumeration value="true"/>
                <xs:enumeration value="false"/>
              </xs:restriction>
            </xs:simpleType>
          </xs:attribute>
          <xs:attribute name="id" type="xs:ID" use="required"/>
        </xs:complexType>
      </xs:element>
      <xs:element name="INSDFeature_partial3" minOccurs="0">
        <xs:complexType xdb:SQLType="PARTIAL3_T">
          <xs:attribute name="value" use="required">
            <xs:simpleType>
              <xs:restriction base="xs:string">
                <xs:enumeration value="true"/>
                <xs:enumeration value="false"/>
              </xs:restriction>
            </xs:simpleType>
          </xs:attribute>
          <xs:attribute name="id" type="xs:ID" use="required"/>
        </xs:complexType>
```

```
    </xs:element>
    <xs:element name="INSDFeature_quals" minOccurs="0">
     <xs:complexType xdb:SQLType="QUALS_T">
      <xs:sequence minOccurs="0" maxOccurs="unbounded">
       <xs:element ref="INSDQualifier"/>
      </xs:sequence>
      <xs:attribute name="id" type="xs:ID" use="required"/>
     </xs:complexType>
    </xs:element>
   </xs:sequence>
   <xs:attribute name="id" type="xs:ID" use="required"/>
  </xs:complexType>
</xs:element>

<xs:element name="INSDInterval" xdb:defaultTable="">
 <xs:complexType xdb:SQLType="INTERVAL_T">
  <xs:sequence>
   <xs:element name="INSDInterval_from" type="integerWithId" minOccurs="0"/>
   <xs:element name="INSDInterval_to" type="integerWithId" minOccurs="0"/>
   <xs:element name="INSDInterval_point" type="integerWithId" minOccurs="0"/>
   <xs:element name="INSDInterval_iscomp" minOccurs="0">
    <xs:complexType xdb:SQLType="ISCOMP_T">
     <xs:attribute name="value" use="required">
      <xs:simpleType>
       <xs:restriction base="xs:string">
        <xs:enumeration value="true"/>
        <xs:enumeration value="false"/>
       </xs:restriction>
      </xs:simpleType>
     </xs:attribute>
     <xs:attribute name="id" type="xs:ID" use="required"/>
    </xs:complexType>
   </xs:element>
   <xs:element name="INSDInterval_interbp" minOccurs="0">
    <xs:complexType xdb:SQLType="INTERBP_T">
     <xs:attribute name="value" use="required">
      <xs:simpleType>
       <xs:restriction base="xs:string">
        <xs:enumeration value="true"/>
        <xs:enumeration value="false"/>
       </xs:restriction>
      </xs:simpleType>
     </xs:attribute>
     <xs:attribute name="id" type="xs:ID" use="required"/>
    </xs:complexType>
   </xs:element>
```

```
      <xs:element name="INSDInterval_accession" type="stringWithId"/>
    </xs:sequence>
    <xs:attribute name="id" type="xs:ID" use="required"/>
  </xs:complexType>
</xs:element>

<xs:element name="INSDQualifier" xdb:defaultTable="">
  <xs:complexType xdb:SQLType="QUALIFIER_T">
    <xs:sequence>
      <xs:element name="INSDQualifier_name" type="stringWithId"/>
      <xs:element name="INSDQualifier_value" type="stringWithId" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="id" type="xs:ID" use="required"/>
  </xs:complexType>
</xs:element>

<!-- This element is a container for record elements.
     Since we will only insert individual records rather than
     sets of records in the BIODQ application, we do not need this.

<xs:element name="INSDSet">
  <xs:complexType>
    <xs:sequence minOccurs="0" maxOccurs="unbounded">
      <xs:element ref="INSDSeq"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

-->

</xs:schema>
```

LIST OF REFERENCES

1. Abiteboul, S., Buneman P., Suciu, D.: Data on the Web: From Relations to Semistructured Data and XML. Morgan Kaufmann Publishers, San Francisco, CA (2000)

2. Adams, D.: Oracle XML DB Developer's Guide, 10g Release 2 (10.2). Oracle (2005). http://download.oracle.com/docs/cd/B19306_01/appdev.102/b14259.pdf

3. Ballou, D., Madnick, S., Wang, R.: Assuring Information Quality. J. Management Information Systems 20(3), 9-11 (2004)

4. Beall, J.: Metadata and Data Quality Problems in the Digital Library. J. Digital Information 6(3), No. 355 (2005)

5. Benson, D.A., Karsch-Mizrachi, I., Lipman, D.J., Ostell, J., Wheeler, D.L.: GenBank. Nucleic Acids Res. 35(Database issue), D21-D25 (2007)

6. Bochmann, G., Hafid, A.: Some Principles for Quality of Service Management. Distributed Systems Engineering J. 4(1), 16-27 (1997)

7. Boeckmann, B., Bairoch, A., Apweiler, R., Blatter, M.C., Estreicher, A., Gasteiger, E., Martin, M.J., Michoud, K., O'Donovan, C., Phan, I., Pilbout, S., Schneider, M.: The SWISS-PROT protein knowledgebase and its supplement TrEMBL in 2003. Nucleic Acids Res. 31(1), 365-370 (2003)

8. Bossa, S.: Montag (August 2007). http://montag.sourceforge.net/index.html

9. Bouzeghoub, M., Peralta, V.: A Framework for Analysis of Data Freshness. In: Proceedings of the International Workshop on Information Quality in Information Systems (IQIS), pp. 59-67 (2004)

10. Buneman, P.: Semistructured Data. In: Proceedings of the ACM Symposium on Principles of Database Systems (PODS), pp. 117-121 (1997)

11. Buneman, P., Davison, S., Hillebrand, G., and Suciu, D.: A query language and optimization techniques for unstructured data. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 505-516 (1996)

12. Calvanese, D., De Giacomo, G., Lenzerini, M.: Modeling and Querying Semi-Structured Data. Networking and Information Systems J. 2(2), 253-273 (1999)

13. Centrum voor Wiskunde en Informatica: MonetDB (August 2007). http://monetdb.cwi.nl/projects/monetdb/Home/index.html

14. Cyran, M.: Oracle Database Concepts, 10g Release 2 (10.2). Oracle (2005). http://download.oracle.com/docs/cd/B19306_01/server.102/b14220.pdf

15. DeSantis, T.Z., Hugenholtz, P., Larsen, N., Rojas, M., Brodie, E.L., Keller, K., Huber, T., Dalevi, D., Hu, P., Andersen, G.L.: Greengenes, a Chimera-Checked 16S rRNA Gene Database and Workbench Compatible with ARB. Appl Environ Microbiol. 72(7), 5069-5072 (2006)

16. Drake, M.: Oracle XML DB White Paper. Oracle (2005). http://download-uk.oracle.com/otndocs/tech/xml/xmldb/TWP_XML_DB_10gR2_long.pdf

17. European Bioinformatics Institute: EMBL Nucleotide Sequence Database (February 2006). http://www.ebi.ac.uk/embl/

18. France Telecom S.A.: Introduction to ASN.1 (August 2007). http://asn1.elibel.tm.fr/en/introduction/

19. Garcia-Molina, H., Ullman, J., Widom, J. Database Systems: The Complete Book. Prentice Hall, New Jersey (2002)

20. International Nucleotide Sequence Database Collaboration: INSDC (July 2007). http://www.insdc.org/page.php?page=home

21. International Nucleotide Sequence Database Collaboration: INSDC Feature Table Definition Document (September 2007). http://www.insdc.org/files/feature_table.html

22. Kulikova, T., Akhtar, R., Aldebert, P., Althorpe, N., Andersson, M., Baldwin, A., Bates, K., Bhattacharyya, S., Bower, L., Browne, P., Castro, M., Cochrane, G., Duggan, K., Eberhardt, R., Faruque, N., Hoad, G., Kanz, C., Lee, C., Leinonen, R., Lin, Q., Lombard, V., Lopez, R., Lorenc, D., McWilliam, H., Mukherjee, G., Nardone, F., Pastor, M.P., Plaister, S., Sobhany, S., Stoehr, P., Vaughan, R., Wu, D., Zhu, W., Apweiler, R.: EMBL Nucleotide Sequence Database in 2006. Nucleic Acids Res. 35(Database issue), D16-D20 (2007)

23. Lee, Y.W., Strong, D.M.: Knowing-Why About Data Processes and Data Quality. J. Management Information Systems 20(3), 13-39 (2003)

24. Lee, Y.W., Strong, D.M., Kahn, B.K., Wang, R.Y.: AIMQ: A Methodology for Information Quality Assessment. Information and Management 40(2), 133-146 (2002)

25. McHug, J., Abiteboul, S., Goldman, R., Quass, D., Widom, J. Lore: A Database Management System for Semistructured Data. ACM SIGMOD Rec. 26(3), 54-66 (1997)

26. Mecella, M., Scannapieco, M., Virgillito, A., Baldoni, R., Catarci, T., Batini, C.: Managing Data Quality in Cooperative Information Systems. J. Data Semantics. Lecture Notes in Computer Science, vol. 2800, pp. 208-232. Springer-Verlag, Berlin Heidelberg New York (2003)

27. Meier, W.: eXist (August 2007). http://exist.sourceforge.net/

28. Mihaila, G., Raschid, L., Vidal, M.E.: Querying "Quality of Data" Metadata. In: Proceedings of the IEEE MetaData Conference, pp. 526-531 (1999)

29. Missier, P., Batini, C.: A Multidimensional Model for Information Quality in Cooperative Information Systems. In: Proceedings of the International Conference on Information Quality (ICIQ), pp. 25-40 (2003)

30. Missier, P., Embury, S., Greenwood, M., Preece, A., Jin, B.: Quality views: capturing and exploiting the user perspective on data quality. In: Proceedings of the International Conference on Very Large Data Bases (VLDB), pp. 977-988 (2006)

31. Mitchell, T.: Machine Learning. McGraw Hill (1997)

32. Mueller, L.A., Solow, T.H., Taylor, N., Skwarecki, B., Buels, R., Binns, J., Lin, C., Wright, M.H., Ahrens, R., Wang, Y., Herbst, E.V., Keyder, E.R., Menda, N., Zamir, D., Tanksley, S.D.: The SOL Genomics Network. A Comparative Resource for Solanaceae Biology and Beyond. Plant Physiol. 138(3), 1310-1317 (2005)

33. Müller, H., Naumann, F., Freytag J.C.: Data Quality in Genome Databases. In: Proceedings of the International Conference on Information Quality (ICIQ), pp. 269-284 (2003)

34. National Center for Biotechnology Information: DATATOOL - NCBI data conversion tool (August 2007). http://www.ncbi.nlm.nih.gov/data_specs/NCBI_data_conversion.html

35. National Center for Biotechnology Information: GenBank Overview (February 2006). http://www.ncbi.nlm.nih.gov/Genbank/

36. National Center for Biotechnology Information: Entrez, The Life Sciences Search Engine (August 2007). http://www.ncbi.nlm.nih.gov/gquery/gquery.fcgi

37. National Center for Biotechnology Information: Expressed Sequence Tags database (Sep 2007). http://www.ncbi.nlm.nih.gov/dbEST/

38. National Center for Biotechnology Information: Index of schema (August 2007). http://www.ncbi.nlm.nih.gov/data_specs/schema/

39. National Center for Biotechnology Information: RefSeq (January 2006). http://www.ncbi.nlm.nih.gov/RefSeq/

40. National Center for Biotechnology Information: The NCBI Handbook (2003). http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?db=Books

41. National Center for Biotechnology Information: The NCBI Help Manual (2006). http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?db=Books

42. National Institute of Genetics: DDBJ-DNA Data Bank of Japan (January 2006). http://www.ddbj.nig.ac.jp/

43. National Library of Medicine: PubMed (July 2007). http://www.ncbi.nlm.nih.gov/sites/entrez?db=PubMed

44. Naumann, F., Freytag J.C., Leser, U.: Completeness of integrated information sources. Information Systems 29(7), 583-615 (2004)

45. Naumann, F., Rolker, C.: Assessment Methods for Information Quality Criteria. In: Proceedings of the International Conference on Information Quality (ICIQ), pp. 148-162 (2000)

46. Naumann, F., Roth, M.: Information Quality: How Good Are Off-The-Shelf DBMS? In: Proceedings of the International Conference on Information Quality (ICIQ), pp. 260-274 (2004)

47. Oracle: Oracle XML DB, 10g (October 2007). http://www.oracle.com/technology/tech/xml/xmldb/index10_2.html

48. Preece, A.D., Jin, B., Pignotti, E., Missier, P., Embury, S.M., Stead, D., Brown, A.: Managing Information Quality in e-Science Using Semantic Web Technology. In: Proceedings of the European Semantic Web Conference (ESWC), pp. 472-486 (2006)

49. Pruitt, K.D., Tatusova, T., Maglott, D.: NCBI reference sequences (RefSeq): a curated non-redundant sequence database of genomes, transcripts and proteins. Nucleic Acids Res. 35(Database issue), D61-D65 (2007)

50. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, San Francisco, CA (1993).

51. Rakotomalala, R.: TANAGRA: a free software for research and academic purposes. In: Proceedings of EGC, RNTI-E-3, vol. 2, pp. 697-702 (2005)

52. Rice, J.A.: Mathematical Statistics and Data Analysis, 2nd edn. Duxbury Press, Belmont, CA (1995)

53. Scannapieco, M., Virgillito, A., Marchetti, M., Mecella, M., Baldoni, R.: The DaQuinCIS Architecture: A Platform for Exchanging and Improving Data Quality in Cooperative Information Systems. Information Systems 29(7), 551-582 (2004)

54. Schmutz, J., Wheeler, J., Grimwood, J., Dickson, M., Yang, J., Caoile, C., Bajorek, E., Black, S., Chan, Y.M., Denys, M., Escobar, J., Flowers, D., Fotopulos, D., Garcia, C., Gomez, M., Gonzales, E., Haydu, L., Lopez, F., Ramirez, L., Retterer, J., Rodriguez, A., Rogers, S., Salazar, A., Tsai, M., Myers, R.M.: Quality assessment of the human genome sequence. Nature 429(6990), 365-368 (2004)

55. Steinmetz, R., Wolf, L.C.: Quality of service: where are we? In: Proceedings of the International Workshop on Quality of Service (IWQoS), New York, USA (1997)

56. Strong, D., Lee, Y., Wang, R.: Data Quality in Context. Communications of the ACM 40(5), 103-110 (1997)

57. Sumner, T., Khoo, M., Recker, M., Marlino, M.: Understanding Educator Perceptions of "Quality" in Digital Libraries. In: Proceedings of the ACM/IEEE-CS Joint Conference on Digital Libraries, pp. 269-279 (2003)

58. Swiss Institute for Bioinformatics and European Bioinformatics Institute: SwissProt (November 2006). http://www.expasy.org/sprot/

59. Tan, P., Steinbach, M., Kumar, V.: Introduction to Data Mining. Addison Wesley (2005)

60. Wang, R.Y., Reddy, M.P., Kon, H.B.: Toward Quality Data: An Attribute-Based Approach. Decision Support Systems 13(3-4), 349-372 (1995)

61. Wheeler, D.L., Barret, T., Benson, D.A., Bryant, S.H., Canese, K., Chetvernin, V., Church, D.M., DiCuccio, M., Edgar, R., Federhen, S., Geer, L.Y., Kapustin, Y., Khovayko, O., Landsman, D., Lipman, D.J., Madden, T.L., Maglott, D.R., Ostell, J., Miller, V., Pruitt, K.D., Schuler, G.D., Sequeira, E., Sherry, S.T., Sirotkin, K., Souvorov, A., Starchenko, G., Tatusov, R.L., Tatusova, T.A., Wagner, L., Yaschenko, E.: Database resources of the National Center for Biotechnology Information. Nucleic Acids Res. 35(Database issue), D5-D12 (2007)

62. World Wide Web Consortium: Extensible Markup Language (XML) 1.0 (Second Edition), W3C Recommendation 6 October 2000 (August 2007). http://www.w3.org/TR/2000/REC-xml-20001006

63. World Wide Web Consortium: XML Schema (August 2007). http://www.w3.org/XML/Schema

64. World Wide Web Consortium: xml:id Version 1.0, W3C Recommendation 9 September 2005 (July 2007). http://www.w3.org/TR/xml-id/

65. World Wide Web Consortium: XQuery 1.0: An XML Query Language, W3C Recommendation 23 January 2007 (August 2007). http://www.w3.org/TR/xquery/

BIOGRAPHICAL SKETCH

Alexandra Martínez was born on December 28, 1978 in San José, Costa Rica. She grew up with her older sister and parents in Heredia, a city close to San José. She graduated from the Escuela Cubujuquí (Cubujuqui Elementary School) in 1990 and from the Colegio Científico Costarricense (Costa Rican Scientific High School) in 1995. She earned her B.S. in computer and information science from the University of Costa Rica in 2000. Upon graduation, she joined ArtinSoft, a leading software migration company in Costa Rica. In 2001, she decided to pursue graduate studies abroad after receiving a fellowship from the École Polytechnique Fédérale de Lausanne, Switzerland, to enroll in the Pre-Doctoral Program in computer science. Upon completion of this program in2002 she moved to Florida, where she continued her Master and Doctoral studies in computer engineering at the University of Florida. She obtained her M.S. in 2006, and her Ph.D. in 2007.

Alexandra has been married to Arturo Camacho since 2002, and they have a lovely daughter, Melissa Maria, who is 18 moths old.