A Metadata Framework for Interoperating Heterogeneous Genome Data Using XML

Kei-Hoi Cheung, Ph.D.\textsuperscript{1}, Aniruddha M. Deshpande, M.D.\textsuperscript{1}, Nicholas Tosches, M.D.\textsuperscript{1}
Sujai Nath, M.D.\textsuperscript{1}, Abha Agrawal M.D.\textsuperscript{1}, Perry Miller, M.D., Ph.D\textsuperscript{1}
Anuj Kumar, Ph.D.\textsuperscript{2}, Michael Snyder, Ph.D.\textsuperscript{2,3}
\textsuperscript{1}Center for Medical Informatics
\textsuperscript{2}Department of Molecular, Cellular and Developmental Biology
\textsuperscript{3}Department of Molecular Biophysics and Biochemistry
Yale University, New Haven, CT

The rapid advances in the Human Genome Project and genomic technologies have produced massive amounts of data populated in a large number of network-accessible databases. These technological advances and the associated data can have a great impact on biomedicine and healthcare. To answer many of the biologically or medically important questions, researchers often need to integrate data from a number of independent but related genome databases. One common practice is to download data sets (text files) from various genome Web sites and process them by some local programs. One main problem with this approach is that these programs are written on a case-by-case basis because the data sets involved are heterogeneous in structure. To address this problem, we define metadata that maps these heterogeneously structured files into a common eXtensible Markup Language (XML) structure to facilitate data interoperation. We illustrate this approach by interoperating two sets of essential yeast genes that are stored in two yeast genome databases (MIPS and YPD).

INTRODUCTION

Advances in molecular biology offer great opportunities for biomedicine and healthcare. These advances play an important role in the following.

1. Understanding disease mechanisms. Genes are the blueprint of life. They encode proteins that serve important functions in our health. Variation in the gene sequence structure (e.g., single nucleotide polymorphism or SNP) can make the encoded proteins function abnormally, thereby causing diseases. For example, sickle cell anemia is caused by a nucleotide sequence mutation that results in the disruption of the normal function of hemoglobin (a protein that has the crucial function of transporting oxygen in the human body). In this case, the sequence variation has profound effects on the folding pattern of the hemoglobin, altering its functionality. Understanding the relationship between gene sequence variation and protein structure (as well as protein expression) has been a major aspect of biomedical research.

2. Disease diagnosis. The chance of curing such diseases as cancers is increased if they are detected early. Genetic screening has become a promising tool that can be used to detect certain diseases including colon cancer and cystic fibrosis when they are still at their early stage. In this technique, the patients are screened for the presence or absence of certain DNA markers, which help predict the likelihood of disease for those patients as well as their offsprings. Biotechnologies such as Polymerase Chain Reaction (PCR) \cite{1} have helped produce a great number of such diagnostic markers.

3. Disease treatment. Once the defective genes that cause a disease are identified, methods can be devised in an attempt to repair or replace these genes or to tailor drug treatment to the specific genetic variation of individual patients (pharmacogenomics \cite{2}). Experimental techniques have been explored to treat diseases at the nucleotide level. Gene therapy, for example, is a technique that is used to replace the faulty gene with a working one, so that the human body can make the correct enzyme or protein and consequently eradicate the root cause of the disease.

The rapid advances in the Human Genome Project (HGP) \cite{3} and genomic technologies such as microarrays \cite{4} have given rise to massive amounts of data populated in a large number of databases that are accessible over the Web. One key feature of the Web is that it allows different Web-accessible databases to be connected to one another through hypertext links. These hyperlinks revolutionized the way information (stored in remote databases) could be linked together via the Internet. This level of database inter-connectivity has already proven very useful since it allows the user to navigate data from one Web site to another related site. According to Karp \cite{5} however, this is not the "Holy Grail" of genome database integration. There are two problems associated with this hypertext linking approach.

1. Item-by-item linking. One problem is that the user has to click on each link one at a time to retrieve related information. This will be a time-consuming and tedious method to collect related information if the amount of data and the number of links involved is large, thereby slowing data collation prior to analysis.

2. Fixed linked fields. The other problem occurs when we attempt to establish links to external data sources. These external links restrict how we can access related data. For example, some genome
databases allow their data entries to be linked via accession numbers (unique object/record identifiers). However, if these numbers are not available in the public interface, there will not be any way to establish links using other fields (such as gene names or gene symbols).

The genome community needs a more flexible and powerful form of database interoperation to facilitate large-scale integrative data analysis [6]. At present, the method of interoperation often adopted involves writing custom programs to integrate structured data files (e.g., tab-delimited text files) downloaded from different Web sites. One problem with this approach is that there are no standard formats and standard vocabularies. Typically, a specific program is needed to parse the specific structure of a data file. This structural heterogeneity increases the number of processing programs, thereby creating a significant problem in software maintenance.

To address this problem, we create a central metadata repository to describe rules that allow heterogeneous data files to be merged into a single integrated data set using eXtensible Markup Language or XML (http://www.xml.com/pub/a/98/10/guide0.html). This paper describes how to create such metadata and how to use XML as a standard format for interoperating genomic data sets obtained from heterogeneous databases.

Because of its simplicity (it is text-based) and self-describing nature (each value is tagged by a descriptive name that is hierarchically structured), XML has been proposed as a standard for exchanging different types of genome data over the Web. For example, the GeneX Gene Expression Markup Language or GeneXML (http://www.ncgr.org/research/genex/genexxml.html) has been proposed to describe gene expression data. The popularity of XML is also indicated by the growing number of XML-related software tools and technologies including Document Object Model or DOM (http://www.w3.org/DOM/) and eXtensible Stylesheet Language or XSL (http://www.w3.org/Style/XSL/). Once the data are merged and represented in XML format, programs (based on the use of XML technologies) can be written to query or process the data. This approach eliminates the need to write multiple conversion programs since the rules for converting individual data files are described centrally as metadata.

METHODS

Fig. 1 shows the key components of our system. Our approach involves defining metadata to specify how to convert and merge heterogeneous (but related) sets of data obtained from different Web-accessible data sources into a common XML format. In particular, we apply our approach to text (ascii) data files since current genomic analyses are based on integrating and processing these text files that can be accessed or downloaded from different genome Web sites. The two most popular formats for such text files are (i) tabular format (with a delimiter such as tab or comma separating the columns) and (ii) tag-value pairs (with a delimiter separating the tag and the value). In our case, we refer these columns or tags as fields.

A. Data Conversion and Merging

Our data interoperation procedure entails the following steps.

1. Data Input. This step involves either (i) fetching a document dynamically given an URL or (ii) retrieving a static document that was previously downloaded from a Web site. Fetching data dynamically ensures data currency. However, this approach may yield poor performances when the datasets to be fetched are large or when the host program that provides the link is changed or broken (e.g., the server may be down). For large datasets that evolve over time, we can download them periodically and store them locally.
2. Conversion into tables. This involves converting heterogeneously structured data sets obtained from individual sources into the corresponding (intermediate) database tables. This conversion is done dynamically. Metadata is defined to identify which fields in the source files correspond to which XML tags (which are represented as columns of the tables). Fields in different files may have the same meaning but different names. In this case, these fields correspond to the same XML tag.

3. Merging into an XML document. Once the source data files are converted into tables, they can be joined based on matching columns (columns that have the same name "field" values). These joined results (which are captured in a single combined table) can then be converted into the target XML document based on the rules that are described in the metadata. This target XML document can then be directed to other Web sites or Web agents that recognize XML-formatted data for further (automatic) processing.

B. Metadata

To map the source file structure to the target XML structure, we need to specify (i) how the fields in one or more source files correspond to the XML tags or Xtags (elements and attributes) in the target (integrated) XML document and (ii) how the XML tags are related to one another. This involves creating three types of metadata as follows.

1. Correspondence between the file structure and the XML structure. This can be represented by the following pair: <FieldFile, XtagDoc>, which specifies which field(s) of the source file (Field-x) corresponds to which Xtag of the target document (Doc-y). Notice that field can be null or blank, which means that the corresponding Xtag does not map to any field. This allows additional information (which is implicit in the source file) to be represented explicitly in the target XML document. For example, the root element of an XML document may not correspond to any file field, but such an additional element can be used to describe generally what kind of information is stored in the source files.

2. Join keys. When inter-linking multiple data files, we need to identify which set of fields (join keys) among multiple source files can be joined together based on comparisons of their values (this is equivalent to the join concept in the relational database). This can be represented by the following pair: <XtagGene, XtagGene-or-many>. This pair indicates which two fields can be used to join between two data files that are related in a one-to-one or one-to-many fashion.

3. Hierarchical relationships of the XML-components. An XML document is basically a set of hierarchically-related XML components including elements, attributes, and their values. We define the following triple: <Parent-Component, Child-Component, relationship-type>, which describes the relationship between any two XML-components (parent component and child component). There are two types of relationships: "element" and "attribute". For example, <orf, "species", "attribute"> means that "orf" (open reading frame) has an attribute "species".

APPLICATION

To illustrate how our approach works, we have applied our approach to interoperable data between two yeast genome Web sites (YPD [7] and MIPS [8]) that include information on essential genes. These genes encode proteins of essential function for yeast cells, without which the yeast cells cannot survive. This gene set is clinically significant, as many essential genes in yeast are homologs of human onco genes and tumor suppressors. For example, the yeast gene HRTI has a human homolog (Elongin C) that is involved in the human tumor-suppressor gene: von Hippel-Lindau (VHL). When mutated, VHL was found to be linked to human kidney cancers. An understanding of these yeast essential genes will, therefore, shed light on disease-causing mechanisms in humans. Different yeast research groups may have identified different yeast essential genes and stored them in different databases. It is also possible that the same yeast gene is identified as essential by one research group but non-essential by another group. It is therefore important to identify such conflicting yeast genes.

In addition, we illustrate how to use a standard vocabulary for our data conversion and interoperation. In particular, we used Gene Ontology or GO [9], a controlled vocabulary that can be applied to describe genomic data for all eukaryotes such as the yeast and the mouse genome. GO also provides an XML structure for representing the vocabulary/ontology. We used this standard XML structure to convert and interoperate the MIPS and YPD data sets.

These two gene list files contain information including gene names and synonyms, names of the open reading frames (ORF), and gene descriptions. Instead of defining our own structure to capture the information, we extend GO's XML structure for describing gene products (e.g., protein and RNA) as well as their properties including molecular function, biological process, and cellular component. This gene product structure captures most of the information present in the yeast gene sets except for gene descriptions and ORF names. In this case, we added an attribute "orf" and a sub-element "description" to the gene product element. Below is an example (El) illustrating how the extended GO XML structure is applied to describe the essential gene HRTI in MIPS and YPD, respectively.

Example E1

MIPS:
<gene_product name="HRTI" org="yeast" orf="YOL133w" dbname="MIPS essential genes">
  <description>
This example illustrates that the two data sets can be joined based on the values of the *orf* or *name* tag. In this case, *orf* was chosen because every yeast gene is associated with an *orf* name but not a standard gene name (i.e., the *name* field can be blank for some genes). In this example, we notice that the synonym (ROC1) for HRT1 is available in YPD but not in MIPS. Also, a more detailed description of HRT1 is found in YPD. Once the two gene lists are converted into this common XML format, they can readily be compared or integrated. In general, two alternative approaches can be used to integrate the data. One approach is to develop rules to merge the data. For example, we can define some rules to concatenate the gene product descriptions from the two data sources. The other approach is not to merge the data. Instead, we treat tags (even those that have the same name) in different data sources distinct. To disambiguate, we can give each tag a prefix that identifies the data source to which the tag belongs.

**IMPLEMENTATION**

Our approach was implemented using client-side Visual Basic (VB) scripts that include an XML parsing engine—DOM (Document Object Model). These client-side scripts including DOM can be interpreted and executed by the Microsoft Internet Explorer (IE). We have tested our scripts with IE-5. The metadata are represented and managed using Microsoft Access 2000. In addition to metadata, the source files are converted into Access tables as described previously. A demo of this approach, as well as the source code, can be downloaded via the following: http://borscht.med.yale.edu/kei_web/xmlint-amia-01.zip. The demo requires Microsoft Access 2000 running on Windows.

**DISCUSSION**

The advantages of interchanging genome data over the Web using XML as compared to the traditional flat file approach include the following.

1. **Self-describing.** XML documents carry a simple but useful structure. Each data value is associated with a tag that describes the meaning of that value. These tags can be related in a hierarchical fashion.

2. **Validation.** Unlike flat files that cannot be validated, rules can be written to specify semantics including how the tags can be named and related (in what order) using the Document Type Definition (DTD) language.

3. **Low entry cost.** The cost of exchanging XML documents or flat files over the Web is low since no expensive software is needed. In addition, there is a large repository of XML-related software available in the public domain (e.g., XML parsers are available for a variety of programming languages including PERL and JAVA).

Despite these advantages, XML documents are larger than their flat file counterparts because each data value is associated with a tag or a pair of start tag and end tag. To process large XML documents (e.g., joining them together based on some common tag values) may yield poor performances. To address this problem, we have used a relational database to speed the conversion process through the use of relational joins. To make better use of the relational approach (e.g., indexing, querying, and data modeling) in XML processing, we are currently exploring ways to efficiently transform XML documents into relational format. One approach is to map the XML structure into an entity-attribute-value (EAV) structure [10] because of their structural similarity (the XML structure can be considered a form of attribute-value structure). Nadkarni et al [11] have demonstrated how to represent and query EAV data using relational databases (e.g., Access and Oracle). Such an EAV approach has been applied to a variety of clinical and biological data that are fundamentally heterogeneous [12]. This approach may allow the speed of processing large amounts of XML-formatted data to be improved. Using XML-based methods (e.g. DOM) alone to query or process large XML documents may not yield desired performance because of the lack of indexing capability. For example, using DOM to join the two XML documents (each consisting of roughly 1000 genes) with the structure shown in E1 based on common tag values (e.g., orf name) takes almost 4 times longer than joining them as relational tables.

To provide a more useful integration of structurally heterogeneous but semantically related data, we need to seek more sophisticated rules for data integration. For example, Cheung et al. [13] have discussed how to map data with different structures (e.g., mapping between fields and tables) and different properties such as units and scaling factors. The authors have also described how various strategies (e.g., union and concatenation) can be used to combine a set of related database elements (e.g., tables and columns) into a single element. We can adapt such strategies to integrate data in different XML documents.

Sometimes, there may be no direct mapping between the source file structure and the target XML structure (there may be no simple mapping rules between these two structures). In this case, a series of intermediate transformations may be required. In other words, the source file may be transformed into an intermediate
XML document, which in turns may be transformed into the target XML document. To this end, it may be appropriate to use eXtensible Style-sheet Transformation (XSLT), which allows one XML document to be translated into another XML document with a different structure.

Acknowledgements

This work was supported in part by NIH grants G08 LM05583 from the National Library of Medicine, R01 CA77808 and R24 DK58876.

References