Management of XML Documents in an Integrated Digital Library*

David A. Smith Anne Mahoney
Jeffrey A. Rydberg-Cox
Perseus Project, Tufts University
E-mail: {dasmith,amahoney,jrydberg}@perseus.tufts.edu

Abstract

We describe a generalized toolset developed by the Perseus Project to manage XML documents in the context of a large, heterogeneous digital library. The system manages multiple DTDs through mappings from elements in the DTD to abstract document structures. The abstraction of document metadata, both structural and descriptive, facilitates the development of application-level tools for knowledge management and document presentation. We discuss the implementation of the XML back end and describe applications for cross citation retrieval, toponym extraction and plotting, automatic hypertext generation, morphology, and word co-occurrence.

1 Introduction

One of the greatest challenges in building and maintaining a large, heterogeneous DL (digital library) is the necessity of managing documents with widely varying encodings and markup practices. Although the World Wide Web has demonstrated the power of simple links among simple documents, the benefits of more highly structured markup have long been understood.[1] The Perseus digital library project has developed a generalizable toolset to manage XML (Extensible Markup Language) documents of varying DTDs (Document Type Definitions); to extract structural and descriptive metadata from these documents and deliver document fragments on demand; and to support other tools that analyze linguistic and conceptual features and manage document layout.

In over ten years of creating and managing SGML and now XML data, we have been greatly helped by the generality and abstraction of structured

*Support for this work was provided by the Digital Libraries Initiative Phase 2, with primary funding from the National Endowment for the Humanities and the National Science Foundation.

[1] For applications of structured markup in large document systems, see Fraser et al. (1999), Karben (1999), and Lubell (1999).

markup, which has allowed us to deliver our content smoothly on a variety of platforms, from standalone CD-ROMs, to custom client-server software, to the World Wide Web. In digitizing historical and scholarly documents, we have also come to appreciate the richness of the implicit and explicit links among printed resources. Our DL system reifies these connections and tries to meet the challenges of automatically generating hypertexts in electronic media.[2] Most often needed in creating a rich hypertext across a digital library are models of the structure of individual documents and descriptions of their content. These models ought to be independent of the particular encodings of those documents. Use of these abstractions allows rapid development of scalable tools for display, linguistic analysis, knowledge management, and information retrieval within the DL system. We describe an engine to leverage the power of XML for this modeling task and some of its applications in building a hypertextual digital library. This document management system is the back end for a production web server that delivers over 2 million pages a week; it went into production early in March, 2000.[3]

2 Compiling Document Information

Despite the success of generalized markup schemes such as DocBook or the scheme developed by the TEI (Text Encoding Initiative),[4] projects encoding documents in different fields will require the services of different DTDs (Usdin and Graham, 1998). As XML use grows, more and more specifications for application domains are being created by various industries and user communities; a well known and extensive list of XML applications includes for example an Astronomical Markup Language, a Bioinformatic Sequence Markup Language, a Weather Observation Markup Format, and Genealogical Data in XML.[5] These communities can not only map information in their domains onto markup elements, but can control the interrelationships of information through the XML content model.

Customizing markup applications eases the encoding of individual documents and often achieves a better fit with their intellectual structures, but it can raise barriers to resource discovery within a digital library. To take a simple example, a programmer wishing to extract all of the book titles mentioned in a collection of documents marked up in varying DTDs may have to look for `<cit>` in some documents and `<title>` in others, whose DTD might use `<cit>` to mean a piece of quoted text. More importantly, a document's structural metadata, which is intended to allow access to logical parts of the document, is

[2] For automatic hypertext generation, see Agosti et al. (1997), Allan (1997), and Kaindl and Kramer (1999).

[3] The Perseus digital library is on the web at `http://www.perseus.tufts.edu`. Further information about Perseus may be found in Smith et al. (2000), Crane (1998b), and Crane (1998a).

[4] Perseus does in fact use the TEI DTDs heavily. One of the principal virtues of this tagset is the extensive documentation (Sperberg-McQueen and Burnard, 1994).

[5] `http://www.oasis-open.org/cover/xml.html`

tied to specific concrete element names. In order to find "chapter 5" in an XML document, the user or system implementor must know something about that document's markup — whether the DTD and encoding conventions favor counting to the fifth `<chapter>` tag or searching for `<div2 type="chapter" n="5">`. Although much work has been done on XML Namespaces to encourage markup reuse and minimize duplication of semantic structures (XMLNS 1999), it is unlikely that all marked up documents will eventually use `<ns:title>` for book titles.

Our system, therefore, allows digital librarians to create partial mappings between the elements in a DTD and abstract structural elements and then produces an index (internally termed a lookup table or LUT) of the elements so mapped. What is encoded as `<div1 type="scene">` in one document and as `<scene>` in another is presented to the system as an abstract, structural "scene". Identifier attributes on XML elements, such as `ID` or `N`, are also indexed, and occurrences of each structure within the document are sequentially numbered. A mapping may also specify that some of the document's content, such as section titles or dates, be incorporated in the index to enable resource discovery and visualization, as described below. Unlike SGML architectural forms, this structural mapping is external metadata and does not require modifying either the document or the DTD.[6] Not only does this system insulate the digital library's user from particulars of markup when browsing documents, but it allows standard abstract citations to be mapped onto documents. In looking up, for example, a request for "book 3, Bekker page 1277a" of Aristotle's *Politics*, the DL system does not need to ascertain that the document at hand encodes that information as `<div1 type="Book" n="3">...<pb ed="Bekker" n="1277a">`. Note also that not all elements in a DTD need be mapped, which allows the digital librarian to concentrate on "interesting" structural features.

After this structural element map is created for the DTD, an unlimited number of documents written in conformance with this DTD can be parsed and indexed by the DL system. For efficiency in later extraction of fragments from the XML document, open elements and other persistent state variables in the parsing process are saved at each indexed node.

In addition to indexing structural metadata, the DL system also extracts descriptive metadata about both the document itself (e.g. its creation date) and its subdocuments (e.g. chapter titles). These metadata are stored in a database instantiation of RDF (the Resource Description Framework) where each tuple is a metadata assertion (RDF 1999). Thus one database row may state that a particular document has the Dublin Core Creator "William Shakespeare", another row will state that that document's Dublin Core Title is "Measure for Measure", and another that it can be addressed by acts, scenes, and lines, or by continuous Through Line Numbers. Since documents have permanent identifiers, digital librarians may add some metadata directly to the database rather than having to encode it in the XML. The metadatabase can also be serialized into XML for exchange with other applications, such as more traditional library

[6] For a good discussion of architectural forms, see DeRose and Durand (1994).

catalogues.

3 Delivering Documents

Currently, the main use of our DL system is for delivery of document sections on the World Wide Web. Users specify the text they wish to read; the DL locates the XML file and associated metadata, determines which portion of the file contains the desired text section, applies appropriate styling rules, and presents the text in HTML form.

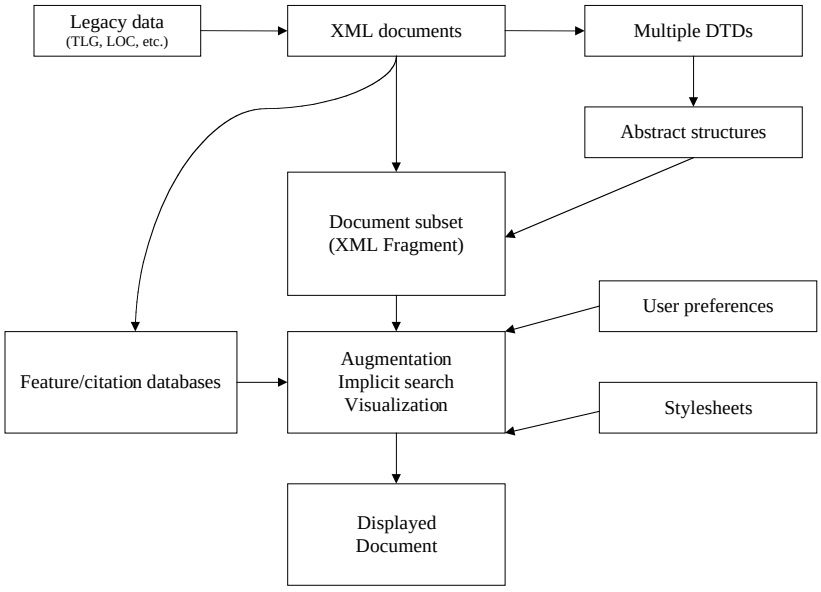

Figure 1: Information flow in Perseus

Users can follow the most direct path into texts in Perseus by looking them up by author, title, and section, according to domain-specific canonical reference schemes. In this case, the first task of the DL display engine, therefore, is to interpret the human-readable citation scheme. We have chosen to map actual electronic documents to abstract bibliographic objects (ABO). Each ABO represents a unit of intellectual content in the digital library.[7] The ABO identifier

[7]In very rough terms, an ABO is a book. More accurately, it is the unit of cataloguing. A single printed book may contain several different literary works, or a work may appear in several volumes. We create an ABO for each work that we expect will be requested by readers

is the key to several of the various metadata tables. Given the ABO, the DL system can determine which concrete XML documents or subdocuments instantiate versions of the desired text. Different versions might include translations into other languages or historical states of the same text; each version is implemented as a separate XML document. We select one version for display, based on user-specified preference information, and give the user a list of other available versions. For example, if a user requests "Matthew 28:5", the DL system looks up the ABO for the Gospel According to Matthew; determines that the original Greek, the Latin Vulgate, and two English translations are available; finds out that the current user prefers Greek; and remembers that the desired section is chapter 28, verse 5.

The DL system also uses the ABO to determine which texts are commentaries on the desired target text. We usually consider a commentary to belong to an ABO, not to a particular version of the text. Many scholarly commentaries, particularly in literary studies, are written with specific reference to the original-language version of a text; we have chosen to make those commentaries available to readers of translations as well as readers of the original language. We distinguish commentaries on an ABO from other texts in the DL that may refer to this ABO; although we will present all available references, those from commentaries are given prominence as being the most relevant to the reader's presumed interests. Presentation of references is discussed further below.

Once the DL system has determined a particular concrete version of the specified ABO, it can retrieve version-specific metadata. These data include the title of this version[8], its language, its layout specification, and the "chunking scheme" for the text. The chunking scheme determines how much of the text is to be presented at once; conceptually, it is analogous to the page size of a printed book. Chunking is expressed in terms of logical structural divisions of the work: chapters, sections, paragraphs, or whatever is appropriate. The DL can apply a default, but the editor of a text may also specify chunking information; there may be a chunking scheme for each citation scheme if the document has more than one. It is convenient if all versions of the same ABO use the same chunking scheme, as this facilitates moving between versions, but our system does not require this.

With all relevant metadata in hand, the DL display engine must open the XML document and extract the portion the reader wishes to see. The engine uses the LUT to find the given citation (for example, the location of Matthew 28:5, in the example given above). The LUT gives not only the byte offset in the XML file, but also a list of the elements open at this point in the text and their attributes. The desired location might fall at the beginning of a chunk or might fall somewhere in the middle; in our example, if the Greek edition of the New Testament is defined to chunk on chapters, Matthew 28:5 is in the middle of the chunk for chapter 28, but Matthew 28:1 would be at the beginning. The chunking routine must locate the beginning and end of the chunk that contains

or cited by other works, and ABOs need not correspond to concrete documents presently in the DL.

[8]It is not unusual for translations to have different titles from original works.

the user's desired section.

Chunks, as noted above, are logical structural divisions of the work, but they need not correspond to container elements in the DTD. In fact, a work may have several different chunking schemes, possibly overlapping.[9] In the actual marked-up text, we implement multiple concurrent hierarchies with empty elements, so as not to require SGML extensions like CONCUR which are neither widely available nor part of the XML standard. Milestones are easy for the markup editor to use, but make chunk extraction a bit more complex. If the desired chunk is implemented by a container element, say for example `<div2 type="chapter" n="28">`, it suffices to extract that entire `<div2>`, and perform closure fixups as described below. If the chunk is the distance between two milestones, say `<milestone unit="chapter" n="28">` and `<milestone unit="chapter" n="29">`, it is possible that the first milestone may be inside one container element and the second may be inside the next of the containers.[10] In such a case, the extracted chunk will overlap with the container elements of the original XML text.

Given the LUT information for the beginning and end of the relevant chunk, then, the extraction routine must read the chunk from the XML file and convert it into a well-balanced XML fragment (XMLFI 1999). This fragment is then wrapped in another element, to become a well-formed XML document, suitable for passing to an XML transformation utility.[11] Our DL currently delivers documents over the Web, so we transform this document into HTML form, but we have experimented with allowing users to request other formats as well, notably Adobe's Portable Document Format and raw XML.

The HTML display is controlled by a template, written mostly in HTML but with place-holders for various display elements and portions of the document. Which template to use is determined by the layout specification metadata record for the concrete XML document. We have chosen to express layout information in HTML primarily so that authors or editors who are not programmers can control the appearance of their documents. Using HTML also allows us to create documents that are idiomatic for the Web, with colors, tables, frames, and the various other features that experienced Web readers expect. That is, although our DL system is more general than an ordinary Web site, and is not tied to the Web as a display medium, it can nonetheless interact natively with the Web in ways that users find intuitive and appealing. As browser support for XML becomes more robust, we expect to exploit XSL (Extensible Stylesheet Language) or other XML styling tools to produce attractive direct XML displays.

[9] Although the DL display engine recognizes a default chunking scheme for each concrete document, the user can request use of a different one.

[10] This may seem unrealistic as applied to the example text, the New Testament, which has a single canonical citation scheme. A clearer example is the book of Psalms, in which containers might be used for the Hebrew numbering scheme, milestones for the Septuagint.

[11] The utility we in fact use is the Copenhagen SGML Tool (CoST), which is written in the mature application extension language Tcl; this puts the full power of a programming language in our hands. As extension mechanisms for XSLT (Extensible Stylesheet Language Transformations) mature, we could equally well use this facility.

The last phase of display processing is the application of the rest of the user's preferences. The first preference was the choice among versions of a text, for example original language or translation; see above. The other preferences govern the appearance of the display and the connections between this text and others in the DL. Users may specify the fonts to be used for languages that do not use the Roman alphabet; we support Unicode, but also support a variety of other fonts for users whose browsers or operating systems do not support Unicode. Users may also indicate which of several automatic searches should be performed, and how the results should be displayed; automatic searches and cross references are discussed further below.

In the end, the user receives a Web page, with standard navigation controls and a text-specific layout, containing a particular fragment of a particular version of a text.

4 Advanced Applications

The advantages of this DL system extend far beyond the practical issues related to delivering documents to end users. The abstraction of structural and descriptive metadata allows us to develop scalable tools for linguistic analysis, knowledge management, and information retrieval within the DL system. The tools in the currently existing DL system include reification of cross-citations of documents within the system, generation of maps with a GIS (Geographic Information System), creation of hypertext, discovery of word co-occurrence patterns, and linkages to morphological analysis.

A great deal of traditional scholarship involves tracking down footnotes and discovering what others have said about a text. The DL system allows us to display links to other texts that cite the document currently being displayed. A simple example is a commentary, which explicitly talks about another text. For example, when a reader views the text of Catullus, the DL system automatically shows notes from E. T. Merrill's commentary on those poems. Much more exciting, however, is the ability to display citations from texts that are not explicitly related to each other. For example, a reader of Matthew 26:5 might wish to know that this text is quoted in James J. O'Donnell's commentary on Augustine's *Confessions*, book 9, chapter 1, section 1. The DL system extracts not only commentary citations but also citations from independent texts, and displays them as active hyperlinks alongside the text being read. This feature shows one benefit that can be gained by taking the time to mark up elements such as citations. Moreover, it becomes more valuable as the DL grows, revealing unexpected links among texts, which scholars might not have been aware of. A reader of Homer might be surprised to find that *Iliad* 8.442 is cited in the *Variorum Edition* of Shakespeare's *Coriolanus*, V.i.

Another module automatically scans any text in the DL for place names. These names are linked to a GIS that allows users automatically to generate a map of the places mentioned in the entire document, in the section currently being displayed, or in a larger logical unit, for example a book or an act. Here,

our abstraction of structural and descriptive metadata is more important than the markup: place names are discovered by an information extraction system after the documents have been indexed. Because the XML back end described above presents all documents to the information discovery module in a uniform way, the GIS programmer need not be aware of the details of the DTDs or of the markup conventions used in the texts.

We have exploited this abstraction in other information extraction systems we have written, one of which is the creation of automatic hypertexts. Important subject terms within the domain area are recognized and linked to other documents within the DL that contain the term. These links are provided automatically for every document when it is displayed. This makes it easy for readers to get fuller information about important ideas, to contextualize unfamiliar vocabulary, and to explore related documents in the DL. Because subject terms are linked to dynamic hypertexts, not to simple glossary or dictionary entries, readers can explore types of material they might not have thought relevant (or even have known of): historical texts, site plans, art works, or the like. Further, as the library expands, new documents appear on these hypertext pages without any additional programming.

In addition to helping users explore large domains, we also have modules that help users explore the smallest details of the language. These details include morphology and word co-occurrence. Every text in a foreign language is passed through a morphological analyzer. The resulting analyses are placed in a database. When foreign-language texts are displayed, each word is checked against this database, and links are generated between words and their analyses. These analyses, in turn, are linked to a suite of dictionaries and grammars, allowing users to read texts in languages they do not yet know well.

Another module that operates at the level of the word identifies words that regularly co-occur. Abstracted indices are used to scan texts and calculate word frequencies and co-occurrence ratios. Highly significant word pairs can be presented along with lexical information, or in independent tabular displays. This sort of collocation information can yield interesting information about common patterns of language usage.[12] For example, in English, collocation data shows that the mutual information score for the words "strong" and "tea" is much higher than the score for "powerful" and "tea". This suggests that it is much more common to speak of "strong tea" than "powerful tea". Collocation data can also provide a quick overview of the sense in which an author uses a word. For example, if the most common collocates of the word "bank" in a collection of texts were words such as "water", "shade", or "cool", we would know that the author probably was writing about rivers rather than financial institutions.

[12]For the theory and practice of word co-occurrence, see Biber (1993), Biber et al. (1998), Church and Hanks (1990), Church et al. (1991), Sinclair (1991), and Smadja (1991).

5 Conclusion

We have described an XML document management system for the Perseus digital library. This system facilitates development of knowledge management applications including those for display, feature extraction, and automatic hypertext generation. Our DL system facilitates development of these and other applications because it releases the application programmer from the task of indexing collections of documents written in multiple DTDs. Because the modules scale as the DL grows, documents in the integrated DL become more valuable than those existing in isolation.

References

M. Agosti, F. Crestani, and M. Melucci. On the use of information retrieval techniques for the automatic construction of hypertext. *Information Processing & Management*, 33(2):133–144, 1997.

James Allan. Building hypertext using information retrieval. *Information Processing & Management*, 33(2):145–159, 1997.

Douglas Biber. Co-occurence patterns among collocations: A tool for corpus-based lexical knowledge acquisition. *Computational Linguistics*, 9(3):531–538, 1993.

Douglas Biber, Susan Conrad, and Randi Reppen. *Corpus Linguistics: Investigating Language, Structure and Use*. Cambridge University Press, Cambridge, 1998.

Kenneth Church, Willian Gale, Patrick Hanks, and Donald Hindle. Using statistics in lexical analysis. In U. Zernik, editor, *Lexical Acquisition: Exploiting On-Line Resources to Build a Lexicon*. Lawrence Erlbaum Associates, Hillsdale, 1991.

Kenneth Church and Patrick Hanks. Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16(1):22–29, 1990.

Gregory Crane. New technologies for reading: The lexicon and the digital library. *Classical World*, pages 471–501, 1998a.

Gregory Crane. The Perseus Project and beyond: How building a digital library challenges the humanities and technology. *D-Lib Magazine*, January 1998b. See `http://www.dlib.org/dlib/january98/01crane.html`.

Steven J. DeRose and David G. Durand. *Making Hypermedia Work: A User's Guide to HyTime*. Kluwer Acadmenic Publishers, Boston, 1994.

B. Fraser, J. Roberts, G. Pianosi, P. Alencar, D. Cowan, D. German, and L. Nova. Dynamic views of SGML tagged documents. In *Proceedings of the*

Seventeenth Annual International Conference on Computer Documentation, pages 93–98, 1999.

Hermann Kaindl and Stefan Kramer. Semiautomatic generation of glossary links: A practical solution. In *Hypertext '99: Returning to Our Diverse Roots*, pages 3–12, 1999.

Alan Karben. News you can reuse: Content repurposing at The Wall Street Journal Interactive Edition. *Markup Languages: Theory & Practice*, 1(1): 33–45, 1999.

Joshua Lubell. Structured markup on the web: A tale of two sites. *Markup Languages: Theory & Practice*, 1(3):7–22, 1999.

John Sinclair. *Corpus, Concordance, and Collocation*. Oxford University Press, Oxford, 1991.

Frank Smadja. Macrocoding the lexicon with co-occurence knowledge. In U. Zernik, editor, *Lexical Acquisition: Exploiting On-Line Resources to Build a Lexicon*. Lawrence Erlbaum Associates, Hillsdale, 1991.

David A. Smith, Jeffrey A. Rydberg-Cox, and Gregory R. Crane. The Perseus Project: A digital library for the humanities. *Literary and Linguistic Computing*, 15(1):15–25, 2000.

C. M. Sperberg-McQueen and Lou Burnard, editors. *Guidelines for Electronic Text Encoding and Interchange*. Text Encoding Initiative, May 1994.

Tommie Usdin and Tony Graham. XML: Not a silver bullet, but a great pipe wrench. *StandardView*, 6(3):125–132, 1998.

XMLNS 1999. *Namespaces in XML*. World Wide Web Consortium, 14 January 1999. See `http://www.w3.org/TR/1999/REC-xml-names-19990114`.

RDF 1999. *Resource Description Framework (RDF) Model and Syntax Specification 1.0*. World Wide Web Consortium, 22 February 1999. See `http://www.w3.org/TR/1999/REC-rdf-syntax-19990222`.

XMLFI 1999. *XML Fragment Interchange*. World Wide Web Consortium, 30 June 1999. See `http://www.w3.org/1999/06/WD-xml-fragment-19990630`.