The XML Framework and Its Implications for Corpus Access and Use

Nancy Ide

Department of Computer Science
Vassar College
Poughkeepsie, NY 12604-0520 USA
ide@cs.vassar.edu

Abstract

The eXtensible Markup Language (XML) (Bray, et al., 1998) is the emerging standard for data representation and exchange on the World Wide Web. The XML Framework includes very powerful mechanisms for accessing and manipulating XML documents that are likely to significantly impact the way annotated corpora are created and accessed. This paper outlines a few of the possibilities.

Introduction

The eXtensible Markup Language (XML) (Bray, et al., 1998) is the emerging standard for data representation and exchange on the World Wide Web. At its most basic level XML is a document markup language directly derived from SGML (i.e., allowing tagged text (elements), element nesting, and element references). As such, translation of an SGML encoded document into XML is relatively trivial. However, various features and extensions of XML make it a far more powerful tool for data representation and access than SGML. The following outlines some of these mechanisms and suggests ways in which they can be used for creation and exploitation of annotated corpora.

XML links

The recommended practice in encoding annotated corpora is to maintain all or most annotations in separate documents, each of which references appropriate locations in the document containing the original data (Ide & Brew, 2000). This strategy yields, in essence, a finely linked hypertext format where the links specify a semantic role rather than navigational options. That is, links signify the location(s) where markup contained in a given annotation document *would appear* in the document to which it is linked. As such, annotation information comprises remote or "stand-off" markup that is virtually added to the document to which it is linked. In principle, the original data could contain no markup at all (or, more likely, markup for gross logical structure only); all markup could be retained in separate documents with links into the original based on offsets.

The standoff scheme, then, requires addressing XML elements, as well as characters and chains of characters within those elements. It also requires that elements and characters can be addressed both within the same document and in other XML documents. XML provides the following linking mechanisms, which are substantially more powerful than the mechanisms provided in SGML, which satisfy these requirements:

- XLink (DeRose, et al., 2000), a mechanism for specifying a link (uni-directional or more complex linking structures) between two or more resources or portions of resources;
- the XML Path Language (XPath) (Clark & DeRose, 1999), an extended addressing syntax that defines a concise notation for element localization in the document tree (as defined by the nesting of elements in the document itself), and allows addressing text fragments within a particular element by providing predicates for manipulating chains of characters;
- XPointer (DeRose, Daniel, & Maler, 1999), which extends XPath syntax to allow addressing points and ranges as well as nodes, locating information by string matching, and use of addressing expressions in URI-references as fragment identifiers.

For example, the Xpath expression `/div/p[2]/s[3]` specifies the third `<s>` (sentence) element within the second <p> (paragraph) element within each `<div>` (text division) element; `/descendant::p` specifies all <p> elements in the document. In addition, Xpath allows addressing text fragments within a particular element by providing predicates for manipulating chains of characters. The expression

```
substring(/p/s[2]/text(),6)
```

selects the string "one would expect that the whole sky would be as bright as the sun, even at night." from the following text:

```
<p><s id="d3p13s4">The difficulty
is that in an infinite static universe
nearly every line of sight would end
on the surface of a star.</s><s
id="d3p13s5">Thus one would expect
that the whole sky would be as bright
as the sun, even at night.</s></p>
```

The expression

```
substring(/p/s[2]/text(), 10, 12)
```

- selects "would expect". Thus the reference is made by specifying (1) the address (absolute or relative) of the element closest to the substring to be referred to, and (2) the substring within this element.

The Xlink mechanism can be used to link corresponding segments of two or more primary texts (As for alignment of text or speech), or to link annotation documents to a base document containing the primary text. For example, in the following, annotation information (e.g., morpho-syntactic information) about a specific token (`<tok>`) is linked to the string of characters in the original text to which it applies:

```
<tok
  xlink:href=
   "substring(/p/s[2]/text(),10,12)">
```

Although this example shows linkage for text, the mechanism provides for linking resources in any medium (audio, video, etc.), which allows for linking speech, external images, video, applets, form-processing programs, style sheets, etc.

In addition to specifying the target location for information in the same or external documents, XLink attributes can be used to specify the role of the link, i.e., how the link should be activated (by hand, or automatically by the browser) and what to do with the target fragment (replace it or insert it into the source document).

In XML, annotated fragments are referenced by the URI (remote or local) of the target resource, and an extended pointer identifying a element and, where necessary, the selected substring of that element's content, as in the following:

```
<tok
  xlink:href=
  "http://www.loria.fr/doc.xml#xptr
  (substring(/p/s[2]/text(), 10, 12))">
```

Annotation resulting from automatic processing (marking of sentence boundaries, tokens, links between parallel texts, etc.) often includes thousands of links to the same external document. Repetition of the document name on every relevant element in an annotation document would obviously significantly multiply its size. XML includes an attribute *xml:base* (Marsh, 2000) that can be used to specify inheritance of an attribute. For example, in the following text:

```
<chunk
  xml:base=
  "http://www.mysite.edu/doc.xml#">
  <tok
      xlink:href="xptr(substring
       (/p/s[2]/text(), 10, 12))"/>
  <tok
      xlink:href="xptr(substring
       (/p/s[2]/text(), 24, 4))"/>
</chunk>
```

the value of the attribute *xml:base* specified on the `<chunk>` element is inherited by the two `<tok>` elements that are its children, and therefore need not be re-specified. The inclusion of `xml:base` in the XML specification ensures that conformant XML processors will handle it (unlike SGML).

XML transformations

The Extensible Style Language (XSL) is a part of the XML framework, consisting of two parts: the best known is the XSL formatting or "style sheet" language; and a powerful tree-traversal language, XSLT (Clark, 1999), that can be used to convert any XML document into another document in any form (e.g., XML, well-formed HTML, plain text, etc.) by selecting, rearranging, and/or adding information to it. The transformed documents may or may not be intended for rendering data on a computer screen, but may be used simply to move data from one computer system or program to another (e.g., to transduce between encoding and/or annotation formats, etc.).

XSLT supports the following kinds of document manipulation:

- selection of elements or portions of element content using the XPath syntax;
- rearrangement or transformation of extracted information (including not only text content but also element names, etc.) in the target document;
- addition of information in the target document.

A suite of documents representing a base text (or texts) and its annotations can be manipulated to serve any application that relies on part or all of its contents. Thus, XSLT is likely to have the most to offer for manipulation of and access to annotated corpora.

XSLT is relatively complex and will not be described in detail here.[1] A short example can provide some idea of the possibilities. Using as input a document containing morpho-syntactic information (e.g., a document containing the fragment in Figure 1[2]), the XSLT document in Figure 2 can be used to create an HTML document that displays a text in "word | lemma | pos" form. When the resulting HTML document is loaded into a browser, it will display the following:

```
It|it|PPER3 was|be|PAST3 a|a|DINT
bright|bright|ADJE cold|cold|ADJE
day|day|NN…
```

```
<?xml version="1.0">
<chunk type="BODY" lang="en"
 xml:base=
"http://www.cs.vassar.edu/~ME/Oen.xcesDoc#">
 <par xlink:href="xptr(substring(//p[1]">
  <s xlink:href="xptr(substring(//p/s[1]">
   <tok type="WORD"
     xlink:href=
     "xptr(substring(//p/s[1]/text(),1,2">
      <orth>It</orth>
      <disamb>
        <base>it</base>
        <msd>Pp3ns</msd>
        <ctag>PPER3</ctag></lex>
      <lex>
        <base>it</base>
        <msd>Pp3ns</msd>
        <ctag>PPER3</ctag></lex></tok>
    <tok type="WORD"
      xlink:href=
      "xptr(substring(//p/s[1]/text(),4,2">
      <orth>was</orth>
      <disamb>
        <base>be</base>
        <msd>Vmis3s</msd>
        <ctag>PAST3</ctag></lex>
      <lex>
        <base>be</base>
        <msd>Vais1s</msd>
        <ctag>AUX1</ctag></lex>
      <lex>
    <base>be</base>
    <msd>Vais3s</msd>
    <ctag>AUX3</ctag></lex>
  <lex>
    <base>be</base>
    <msd>Vmis1s</msd>
    <ctag>PAST1</ctag></lex>
  <lex>
    <base>be</base>
    <msd>Vmis3s</msd>
    <ctag>PAST3</ctag></lex></tok>…
```

Figure 1 : Fragment of a cesAna document

[1] Full documentation is available at http://www.w3.org/TR/xslt.

[2] Note that this document, encoded according to the xcesAna specifications (Ide, Bonhomme, & Romary, 2000), contains full sementation and annotation information, including full morpho-syntactic specifications for all potential annotations and the results of automatic disambiguation.

```
<xsl:stylesheet version="1.0"
   xmnls:xsl=
   "http://www.w3.org/1999/XSL/Transform">

<xsl:template match= "/">
     <html>
       <body>
           <xsl:apply-templates/>
       </body>
     </html>
</xsl:template>

<xsl:template match="//par"/>

  <xsl:for-each select="//tok"/>
     <xsl:value-of select="orth"/>
     <xsl:text>|</xsl:text>
     <xsl:value-of select="disamb/base"/>
     <xsl:text>|</xsl:text>
     <xsl:value-of select="disamb/ctag"/>
  </xsl:for-each>
</xsl:template>

</xsl:stylesheet>
```

Figure 2 : XSLT document to create HTML output

The XSLT script in Figure 2 could be modified to produce output in any desired form, or to produce another XML document containing the merged text and annotation documents. Similarly, XSLT can be used to produce concordances, paired sentences or words from a parallel text, or even a web document that displays the orthographic representation of a text and provides the audio rendition when the word is clicked on, etc. XSLT can also be used to implement an inheritance mechanism over the document tree[3]; for example, Ide, Kilgarriff, & Romary (2000) show how XSLT can implement inheritance mechanism for lexical information.

XML Schemas

The XML Schema definition language (Thompson, et al., 2000; Biron & Malhotra, 2000) enables document creators to constrain and document the meaning, usage and relationships of the constituent parts of XML documents: datatypes, elements and their content, and attributes and their values. Schemas can also be used to provide default values for attributes and elements. As such, XML schemas provide means to define an abstract data model for a class of documents. While duplicating (or making explicit) some of the capabilities provided by XML DTDs, they significantly extend their power and provide for much tighter validation of document form and content.

XML schemas have considerable implications for development of annotated corpora. The following lists only a few possibilities for the application of XML schemas:

- different attribute declarations and/or content models can apply to elements with the same name in different contexts. This allows for more tightly constrained content models than possible with DTDs. For example, names in headers (names of authors, etc., consisting of the usual "first name", "last name" elements) and names in the text ("named entities") should have different content models and attributes in order to provide for tight validation of form in each context. For example, in the SGML-based CES (Ide, 1998a, b), header elements are prefixed with "h." so that names in headers are tagged with `<h.name>`, whose content model is different from that of the `<name>` element that can appear in the body of the text. This strategy is effectively a "kludge" to overcome the fact that SGML provides no scoping capabilities. XML schemas, building on definitions using XML Namespaces (Bray, Hollander, & Layman, 1999), solves this problem. Thus it is possible to avoid the invention of variant element names while retaining the ability to constrain content and attributes based on context.
- equivalence classes can be defined for groups of elements and/or attributes, indicating that they may be used in the same ways as defined for a particular named element ("the exemplar"). For example, the CES makes extensive use of parameter entities to group together elements that behave identically. For example, phrase-level elements (i.e., elements that can appear within but not outside paragraphs or paragraph-like elements, such as name, num, etc.) are grouped using the parameter `%phrase.seq`, so that all paragraph-level elements can include this class in their content models. Again, this is a work-around for the fact that equivalence and inheritance of properties is not expressible in SGML. Similarly, groups of attributes are defined in all CES DTDs, as in the cesAna DTD fragment given in Figure 3. In the XML version of the CES (XCES) (Ide, Bonhomme, & Romary, 2000), this is replaced by the schema in Figure 4.
- attribute or element values, or combinations of attribute and element values, can be constrained to be unique. That is, it is possible to indicate in a computational lexicon that only one entry can be defined with the value of a given word form as its content (or the content of one of its child elements), that only one paragraph can have an attribute indicating that it is the 23rd, or in general that a given key appears only once in a document. Similarly, we can ensure that only one disambiguated form is given for each token in an annotation document, or only one correspondence for a given sentence in an alignment document. Obviously, this is useful for error detection and prevention.
- dependencies can be established based on values of elements or attributes. This has similar benefits for error detection in creating annotated corpora: nouns can be prevented from being assigned a tense, tokens whose *type* attribute has the value PUNCT can be specified to include only `<orth>` elements containing specific characters, etc. More generally, annotation labels (e.g., pos indicators) used in an annotation document can be specified elsewhere, and element content can be constrained to these values only; for example, to constrain the values of the `<msd>` element in an XCES annotation document to the EAGLES morphosyntactic specifications (Monachini & Calzolari, 1996), the following could be specified:

[3] See also Erjavec *et al.* (2000)

```
<element name="msd">
    <simpleType base="string">
      <enumeration value="Pp3ns"/>
      <enumeration value="Vmis3s"/>
      <enumeration value="Vais3s"/>
      <enumeration value="Vmis1s"/>
      ...
    </simpleType>
  </element>
```

```
<!ENTITY % a.global '
        id              ID          #IMPLIED
        n               CDATA       #IMPLIED
        xml:lang        CDATA       #IMPLIED
        lang            IDREF       #IMPLIED'  >

<!ENTITY % a.ana '%a.global;
        type            CDATA       #IMPLIED
        wsd             CDATA       #IMPLIED'
        >

<!ELEMENT cesAna        (cesHeader?, chunkList)>
<!ATTLIST cesAna        %a.ana;
        doc             CDATA          #IMPLIED
        version         CDATA          #REQUIRED>
```

Figure 3 : cesAna DTD fragment for global attributes

```
<schema>
   <attributeGroup name="globalAtt">
     <attribute name="id" type="ID"
           maxOccurs="1" minOccurs="0"/>
     <attribute name="n" type="NMTOKEN"
           maxOccurs="1" minOccurs="0"/>
     <attribute name="lang" type="IDREF"
           maxOccurs="1" minOccurs="0"/>
   </attributeGroup>
   <attributeGroup name="anaAtt">
     <attribute name="type" type="string"
           maxOccurs="1" minOccurs="0"/>
     <attribute name="wsd" type="string"
           maxOccurs="1" minOccurs="0"/>
   </attributeGroup>
   <element name="cesAna">
     <complexType>
      <element name="cesHeader" minOccurs="0"/>
      <element name="chunkList" minOccurs="1"/>
      <attributeGroup ref="globalAtt"/>
      <attributeGroup ref="anaAtt"/>
      <attribute name="doc" maxOccurs="1"
           minOccurs="0"/>
      <attribute name="version" maxOccurs="1"
           minOccurs="1"/>
    </complexType>
  </element>
</schema>
```

Figure 4 : XCES schema for global attributes

Conclusion

This paper outlines some of the potential uses of the mechanisms provided within the XML framework for the creation and use of annotated corpora. As the XML standard develops, more ways to exploit XML will no doubt emerge. For instance, it is likely that corpora and their annotations will be distributed over servers throughout the world in the future, rather than being maintained at a single site; XML is explicitly designed to handle distributed data. Also, XML's provision for accessing part or all of multiple DTDs from a single document provides an elegant means to represent and manipulate standoff annotation documents.

XML is an international standard that will become the base of information exchange and access over the World Wide Web. As we develop annotated corpora that are intended to be maximally usable and reusable for multiple applications, it makes sense to exploit the XML framework to ensure the compatibility and flexibility our applications (and future applications we cannot yet envision) require.

Acknowledgements

The author gratefully acknowledges Laurent Romary and Patrice Bonhomme for their contributions to the ideas in this paper, and for providing the examples of linkage and the use of XML schemas.

References

Biron, P. & Malhotra, A., 2000. XML Schema Part 2: Datatypes. W3C Working Draft, 25 February 2000. http://www.w3.org/TR/xmlschema-2/.

Bray, T., Hollander, D., Layman, M., 1999. Namespaces in XML. World Wide Web Consortium Recommendation, 14 January 1999. http://www.w3.org/TR/REC-xml-names/.

Bray, T., Paoli, J., Sperberg-McQueen, C.M. (eds.), 1998. Extensible Markup Language (XML) Version 1.0. W3C Recommendation. http://www.w3.org:TR/1998/REC-xml-19980210.

Clark, J. (ed.), 1999. XSL Transformations (XSLT). Version 1.0. W3C Recommendation. http://www.w3.org/TR/xslt.

Clark, J. and DeRose, S., 1999. XML Path Language (XPath). Version 1.0. W3C Recommendation. http://www.w3.org/TR/xpath.

Clark, J., 1999. XT Version 1991105. http://www.jclark.com/xml/xt.html

DeRose, S, Maler, E., Orchard, D., Trafford, B. (eds.), 2000. XML Linking Language (XLink). W3C Working Draft, 21 February 2000. http://www.w3.org/TR/xlink.

DeRose, S., Daniel, R., & Maler, E., 1999. XML Pointer Language (XPointer). W3C Working Draft, 6 December 1999. http://www.w3.org/TR/xptr.

Erjavec, T., Evans, R., Ide, N., Kilgarriff, A., 2000. The CONCEDE model for Lexical Databases. In *Proceedings of the Second International Language Resources and Evaluation Conference*, Paris: European Language Resources Association.

Ide, N., 1998a. Encoding Linguistic Corpora. In *Proceedings of the Sixth Workshop on Very Large Corpora*, 9-17.

Ide, N., 1998b. Corpus Encoding Standard: SGML Guidelines for Encoding Linguistic Corpora. In *Proceedings of the First International Language Resources and Evaluation Conference*, Paris: European Language Resources Association, 463-70.

Ide, N., Bonhomme, P., & Romary, L., 2000. XCES: An XML-based Encoding Standard for Linguistic Corpora. In *Proceedings of the Second International Language*

Resources and Evaluation Conference. Paris: European Language Resources Association.

Ide, N. & Brew, C., 2000. Requirements, Tools, and Architectures for Annotated Corpora. In *Proceedings of the EAGLES/ISLE Workshop on Meta-Descriptions and Annotation Schemas for Multimodal/Multimedia Language Resources and Data Architectures and Software Support for Large Corpora* (this volume). Paris: European Language Resources Association.

Ide, N., Kilgarriff, A., Romary, L., 2000. A Formal Model of Dictionary Structure and Content. In *Proceedings of EURALEX'00*, to appear.

Marsh, J., 2000. XML Base (XBase). W3C Working Draft 21-February-2000. http://www.w3.org/TR/xmlbase.

Monachini, M. & Calzolari, N., 1996. Synopsis and Comparison of Morphosyntactic Phenomena Encoded in Lexicons and Corpora: A Common Proposal and Applications to European Languages. EAGLES Report EAG-CLWG-MORPHSYN/R. http://www.ilc.pi.cnr.it/EAGLES96/morphsyn/.

Thompson, H., Beech, D., Maloney, M. Mendelsohn, N., 2000. XML Schema Part 1: Structures. W3C Working Draft, 25 February 2000. http://www.w3.org/TR/xmlschema-1/.