The XML Framework and Its Implications for the Development of Natural Language Processing Tools

Nancy IDE
Department of Computer Science
Vassar College
Poughkeepsie, New York, USA 12604-0520
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 development of tools for processing natural language and annotated corpora.

Introduction

All language processing applications, including machine translation, information retrieval and extraction, text summarization, user/machine dialogue systems, and speech understanding and synthesis, manipulate language data represented in some electronic format. Some applications (e.g., machine translation, summarization, speech understanding) process streams of data more or less sequentially, while others (e.g., retrieval and extraction) rely more heavily on search and access over large bodies of data. In either case, processing exploits the markup in the data to assist in the analysis. For example, in textual data, markup for logical structure (e.g., section, paragraph, and sentence boundaries, etc.) provides essential information for any language processing task. In addition, markup identifying terms, foreign words, names, dates, etc. can be exploited for tasks such as machine translation and information retrieval, while identification of titles, footnotes, and other extra-textual matter can be used to limit the data to be searched. Because the data that will be analyzed by language processing applications in the future will consist largely of documents delivered over the World Wide Web, the markup format these applications process will be XML.

The language processing community also *creates* text and speech data for training statistical language processing algorithms. The cost of creating annotated data can be very high, both in direct financial terms and in terms of the cost of allocating skilled labor. So funders, whether public or commercial, have come to expect that the cost of resource creation will be amortized over multiple research and development efforts. Such reusability demands the use of standardized, non-proprietary encoding formats for data interchange and to enable easy human-readable display and access to data. For the applications we are now beginning to develop, these formats must support multi-lingual, multi-media, and multi-modal data, as well as linkage among them.

As an international standard, the eXtensible Markup Language (XML) (Bray, *et al.*, 1998) is the obvious basis for a standardized encoding format, and is or will be used by several language processing projects (e.g., LT XML[1], ATLAS[2], XCES[3], ANC[4]). At its most basic level

[1] McKelvie, Brew, and Thompson, 1998.

XML is a document markup language directly derived from SGML (i.e., allowing tagged text (elements), element nesting, and element references). However, various features and extensions of XML make it a far more powerful tool for data representation and access than SGML, including means for complex linkage within and between documents, easy data transformations using the XML Transformation Language (XSLT) (Clark, 1999), constraint and validation of markup using XML Schemas (Thompson, *et al.*, 2000; Biron & Malhotra, 2000), and display, manipulation, and search of data via the World Wide Web.

This paper provides an overview of the most important XML mechanisms and suggests how they may impact the design of language processing tools. The focus here is on the use of XML for the creation and annotation of text and speech data; however, we also consider some of the capabilities for search and retrieval from XML-encoded documents.

1 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 may contain no markup at all (or, more likely, markup for gross logical structure only); all markup can be retained in separate documents with links into the original based on offsets.

The standoff scheme, then, requires addressing elements within the original document, 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 other documents. XML provides the following linking mechanisms, 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 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
```

[3] Ide, Bonhomme, and Romary, 2000.

```
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 Xlink mechanism can be used to link corresponding segments of two or more primary documents (as for alignment of text or speech), to link annotation documents to a base document containing the primary data, or, more generally, to link resources in any medium (audio, video, etc.). This 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).

2 XML transformations

The Extensible Style Language (XSL) is a part of the XML framework, consisting of two parts: 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 or documents 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 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, from one or more XML documents;
- 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 document (or documents) 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 impact on the design of language processing tools.

Several projects have developed and implemented language processing tools and tool architectures intended to facilitate flexibility and reusability: for example, MULTEXT (Ide & Véronis, 1994), LT XML (McKelvie, Brew, & Thompson, 1998), GATE (Cunningham, Wilks, & Gaiauskas, 1996), ATLAS (Bird, *et al.*, 2000). While each of these systems is slightly different, they all implement a modular, "plug-and-play" tool architecture based on a three-layered design: one for physical storage representation; one to translate to and from the physical storage representation to one or more internal formats, and an API to enable application development. In addition, all assume SGML or (in the more recently developed systems) XML as the physical representation, together with the use of the stand-off strategy for annotation. The SGML or XML documents containing the data and its annotations are typically transduced into some internal format used by the tools; at any stage in the processing, the results may be transduced back into SGML or XML As a powerful language for selecting from one or several documents and transducing the data into other formats, XSLT provides the means to enable the import and export of data from and to XML.

Of course, XSLT can be used with the documents resulting from processing by tools to deliver the data in any desired format. Although space prevents a full description of XSLT, which is relatively complex, 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[5]), 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 an xcesAna document

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 data and annotation documents (see [www.cs.vassar.edu/XCES] for some more sophisticated examples). 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[6]; for example, Ide, Kilgarriff, & Romary (2000) show how XSLT can implement inheritance mechanism for lexical information.

```
<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

3 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.

xcesAna specifications (Ide, Bonhomme, & Romary, 2000), contains full segmentation and annotation information, including full morpho-syntactic specifications for all potential annotations and the

XML schemas have considerable implications for the creation of annotated data. 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, building on definitions using XML Namespaces (Bray, Hollander, and Layman, 1999). 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.
- 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").
- 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), only one paragraph can have an attribute indicating that it is the 23rd, only one disambiguated form is given for each token in an annotation document, or only one correspondence for a given item 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 annotation documents: 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. In addition, annotation labels (e.g., POS indicators) can be given annotation document can be specified elsewhere, and element content can be constrained to these values only.

Conclusion

This paper outlines some of the potential uses of the mechanisms provided within the XML framework for the creation and use of annotated text and speech data. Because XML is an international standard that is becoming the base of information exchange and access over the World Wide Web, high-end language processing applications intended to extract and manipulate information from diverse sources will necessarily handle XML. It is to our advantage to exploit the XML framework to our greatest advantage, and to ensure compatibility of the data we create with the emerging standard.

Acknowledgements

The author gratefully acknowledges the contributions of Laurent Romary, Patrice Bonhomme, and Chris Brew to the ideas and examples in this paper.

References

Bird, S., Day, D., Garofolo, J., Henderson, J., Laprun, C. Liberman, M., 2000. ATLAS: A Flexible and Extensible Architecture for Linguistic Annotation. In *Proceedings of the Second International Language Resources and Evaluation Conference*. Paris: European Language Resources Association, 1699-1706.

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., 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.

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/.

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.

Cunningham, H., Wilks, Y., Gaizauskas, R., 1996. GATE -- a General Architecture for Text Engineering. In *Proceedings of the 16th International Conference on Computational Linguistics, COLING-96*, Copenhagen, Denmark, 1057-1060.

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.

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*. Paris: European Language Resources Association, 1-6.

Ide, N., & Véronis, J., 1994. MULTEXT: Multilingual Text Tools and Corpora. In *Proceedings of the 15th International Conference on Computational Linguistics, COLING'94*, Kyoto, Japan, 588-92.

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, 825-30.

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

Macleod, C., Ide, N., Grishman, R., 2000. Progress Report on the American National Corpus. In *Proceedings of the Second International Language Resources and Evaluation Conference* (to appear). Paris: European Language Resources Association, 831-35.

McKelvie, D., Brew, C., & Thompson, H. 1998. Using SGML as a Basis for Data-Intensive Natural Language Processing. *Computers and the Humanities* 31:5, 367-388.

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/.