A Method for Automating Text Markup

Rashmi K. Iyengar and R. M. Malyankar*
Department of Computer Science and Engineering
Box 875406, Arizona State University,
Tempe, AZ 85287-5406.
E-mail: rmm@acm.org
URL: http://www.eas.asu.edu/~gcss/research/navigation/

Abstract

Markup languages based on XML are increasingly popular, and languages for other formats such as RDF are under active development. One of the problems involved in converting legacy documents to use XML or other markup formats is the insertion of tags into the document and the consequent rearrangement of text required when markup is added to an existing, un-marked-up document. This paper describes a method for automating part of the process of marking up such legacy documents. The approach is designed for semi-structured text documents, for example, technical documentation and narrative descriptions.

1. Introduction

Markup languages based on XML (Extensible Markup Language [W3C, 2000]) are increasingly being used on the World Wide Web. Other formats such as RDF (Resource Description Framework [W3C, 1999]) and DAML (DARPA Agent Markup Language [Horrocks et al., 2001]) are also gaining acceptance. Unfortunately, converting existing documents to use such markup formats can be a highly labor-intensive task, especially for text documents which were originally written by humans (compared to those which can be generated automatically from a database). Tools to help this conversion process would be useful. Such editors must be able to work with generalized markup, since there are numerous markup languages under development for different domains.

One part of our current Digital Government project consists of developing an XML-based markup language (MIML - Maritime Information Markup Language) for maritime information. MIML is intended to be used for multiple purposes and applications within the domain of maritime information, ranging from database interoperability to information management. MIML (in a future deployment edition) is expected to see use within the context of a envisaged project for information management for waterborne transport [Spalding and Pirzada, 2001, Spalding et al., 2002]. One application of MIML will be marking up text documents used by mariners with a view to facilitating information extraction and presentation. The development and structure of MIML is described in [Malyankar, 2002]. This paper describes a semi-automated editor we are developing to facilitate this task.

The editor described here is designed for XML markup of text that uses a somewhat restricted vocabulary and restricted sentential structure, but which is written in natural language. The application domain for our Digital Government project has large quantities of legacy text that will need to be marked up at some point of time. Much of the text in this domain is semi-structured, in the sense that the same sentential forms are used with slight variations (and different place names, coordinates, compass directions,

*Corresponding author

```
<xs:element name="scope">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="from" /> <xs:element ref="to" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="from">
  <xs:complexType>
    <xs:attribute name="name" type="xs:string" default="" />
    <xs:attribute name="latitude" type="xs:string" default="" />
    <xs:attribute name="longitude" type="xs:string" default="" />
  </xs:complexType>
</xs:element>
```

Figure 1: The 'scope' and 'from' elements from a sample MIML schema

etc.) to describe different circumstances, places, features, etc., and therefore fits the requirements for a restricted vocabulary and restricted sentential structure mentioned earlier. We believe this observation may be true for much other 'professional' text documentation as well. ('Professional documentation' is used in the sense of engineering documentation, in particular, operation manuals, installation guides, etc. It also covers similar documents in non-engineering fields, e.g., government regulations and reports, etc.) Such documents are, by design, written using standard vocabularies and structure (which are usually specific to the application area), and therefore seem amenable to the approach adopted in this tool. Further, as has been pointed out elsewhere in this paper, the sheer volume of this material will impose a real need for even simple automation of markup.

MIML is used to implement text extraction and condensation capabilities in a prototype demonstration (described in [Malyankar, 2001]). MIML is currently being used in the prototype to markup a target document (Chapter 4 of Volume 7 of the *Coast Pilot*, a text document describing features of the United Sates coastline) for subsequent query (using an implementation of XML Query) and information extraction. Note that markup does not consist of simply placing tags around sentences in the text, but involves a limited degree of "understanding", analysis, and rearrangement. Most such conversion tasks are essentially simple, but, considering the high degree of repetitiveness and attention to detail required, are prone to error when done by a human. Such features make these tasks ideal candidates for automation.

2. Outline of Solution

XML schemas [W3C, 2001a] describe the contents of XML documents. They provide definitions of markup language elements, attributes, and data types. Very briefly, as compared to DTDs, XML schemas allow more detail about element structure and constraints. Figure 1 is an example of XML schema elements for the "Coast Pilot". Shown are `<scope>` and `<from>` elements (the first describes the geographical coverage of other document elements in linear terms, and the second can describe either a starting point for a segment, or a location using either its name, or latitude/longitude coordinates, or both).

The XML Schema recommendation [W3C, 2001a, W3C, 2001b] allows `<annotation>` schema components in XML schemas; these elements can contain `<documentation>` elements (supposed to hold text intended for human users, explaining the underlying schema component) or `<appinfo>` (application information) elements (supposed to hold material which is meaningful to software). We use the `<appinfo>` element to contain hints for markup of text. Currently, three types of hints are available:

- *Regular expressions* denoted by the tag `<regex>` for recognizing text fragments and instantiating

local variables for deeper markup processing. A regular expression is a sequence of characters interpreted by the expression matching engine as corresponding to another sequence of characters. For example the string "a*" may be interpreted as matching a string of length 1 or higher, beginning with the letter 'a'.

- *Rearrangements* denoted by `<rule>` elements, for constructing a rearranged text fragment from previously recognized text, and passing it to elements that may be described elsewhere in the schema file. Rearrangements are needed because some components may be part of multiple other components. It is convenient to specify rules for recognition of an element in only one place (e.g., latitude may appear in numerous places in a schema, but it is awkward and error-prone to place the expression that recognizes the text string for a latitude in every such place in the schema). When elements are 'nested', the text corresponding to the smaller component may be only a part of the text for the larger components, and in a different place in each such larger component, necessitating that only a (possibly re-arranged) part of the current text fragment be supplied to the recognizer for the smaller component.

- *Attribute values*, denoted by `<attrval>`, for instantiating attribute values from recognized text.

These three categories of hints appear to be sufficient for recognizing text that can be described by regular expressions, and for a limited amount of text rearrangement. Practically speaking, this means that the class of fragments that can be recognized is the class of 'regular languages' in the Chomsky hierarchy, which in turn means that significant natural language understanding capability requires significant extension to the recognition component, especially by the use of other forms of description.

The editor is implemented in Java as a Protégé plugin. Protégé is a knowledge engineering tool developed by the Stanford Medical informatics group [Grosso et al., 1999, Noy et al., 2000]. It can be used to create and visualize ontologies, as a knowledge base editor, for conversion of knowledge bases to XML or RDF (and shortly, DAML), etc. Our project has used it for ontology construction and ontology processing. Because one principle guiding the design of MIML is the maintenance of a well-defined relationship between MIML tags and the underlying ontologies [Malyankar, 2002], and because we would like, in the near future, to mark up text directly from an ontology/knowledge base (direct markup is explained below), it was decided to integrate the editor described here with the Protégé system.

Figure 2 shows the first version of the editor. The editor is active when the *XMLHelper* tab is selected. The upper left panel is intended to hold the ontology (this can be loaded with Protégé); the upper right and the lower left panels hold the text of the schema file and a tree-based depiction of the schema, respectively; the lower right panel holds the text being marked up. The buttons above the panels on the right are for loading/saving the schema and target text, and for marking up the target text. Markup is 'direct' as far as the user is concerned, i.e., all the user need do is link the element and text fragment in question and then click a button (informally, it amounts to the user 'stating' "mark up *this* with *that*", as opposed to inserting element tag pairs one by one everywhere they are needed with keyboard and mouse). Markup is done by selecting (with the mouse) a schema element in the lower left panel and the text fragment to be processed in the lower right panel, and then clicking the *markup* button. Tags are automatically inserted at the appropriate places, attribute values are initialized (where values are available in the target text), etc. The target text can also be directly edited by the user.

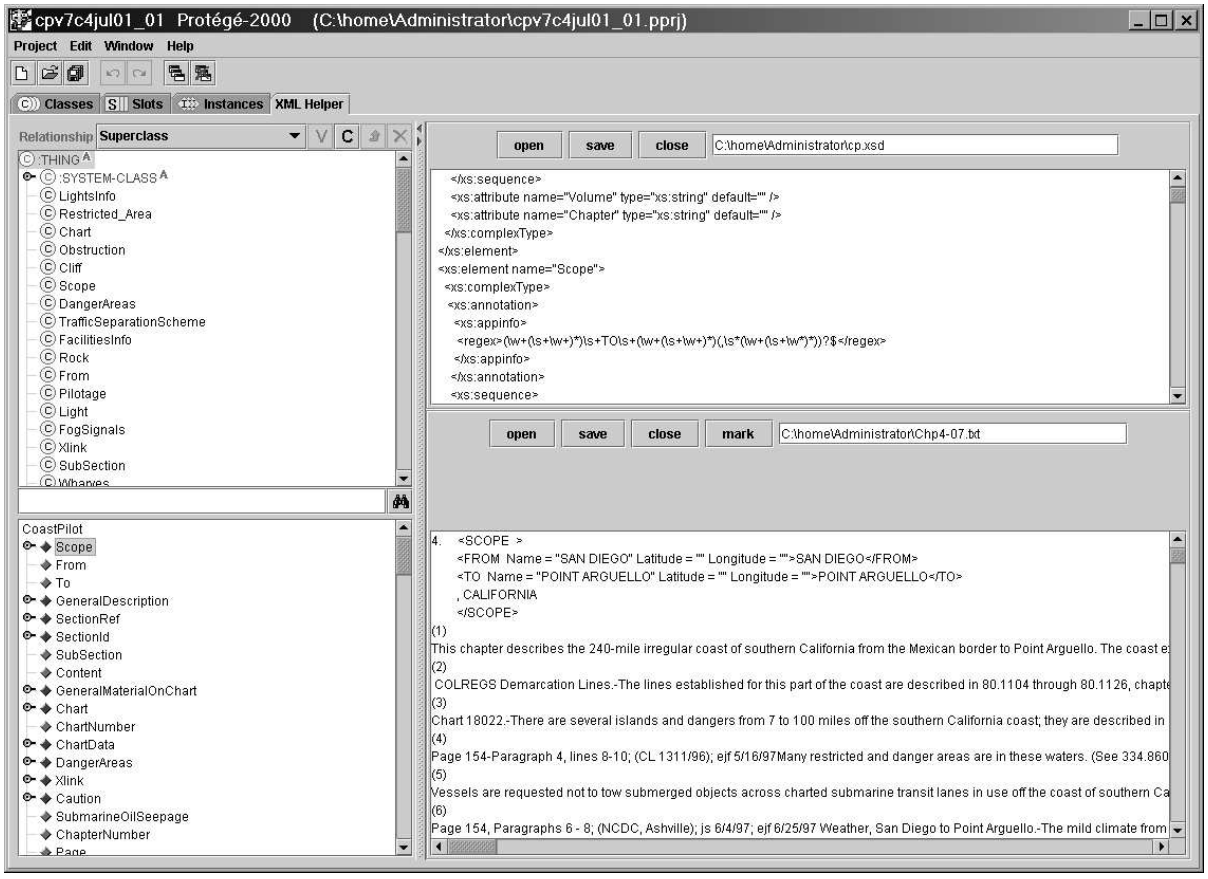

Figure 2: Markup tool showing an XML schema in text and tree forms and the file being marked up

3. Related Work

Markup editors and annotation tools have been in existence for a while, and more are under active development (especially for RDF and DAML+OIL), but most of the current versions appear to depend on manual markup, i.e., require the user to place tags and rearrange text using ordinary text editing techniques — which might be at best mouse- and menu-based. Inserting (a portion of) markup into target documents using text-processing software such as Perl is an obvious idea, and has almost certainly been used in one-off and special-purpose contexts, but we are unaware of any 'recognition-based' tools or tools that do schema-based or ontology-based markup, as has been outlined here. While this may seem a minor help, it should be extremely useful in practice, due to the volume and tediousness of the markup task. In our domain, for example, place names, geographical scopes, and coordinates appear in very many places, and any method of automating markup of even these minor elements is expected to be of significant help to a human.

4. Limitations and Future Work

The editor is not intended to fully automate the markup process, but as an aid for human markup. The utility of the automated recognition facility depends greatly on the nature of text being marked up and the nature of the 'description' (in the first version of the editor, 'regular expressions', which recognize only a restricted subset of natural language). Extension to more capable parsers and recognizers is underway, with

more expressive means of description and wider natural language understanding capabilities. Also, devising the regular expressions currently needs programming expertise; simpler means of describing the text to be marked up are being investigated. Rearrangement of text consists of only limited rewriting capability; using rewrite rules in the hints is another direction of research. Other extensions concern active processors in the hints, and multiple, richer, forms of description, in addition to regular expressions.

The primary reason for selecting XML for this project in preference to RDF or DAML+OIL is the maturity of XML technology compared to these and other languages. On the other hand, RDF is even more closely tied to ontologies and knowledge representation than XML, and DAML+OIL is designed as an extension to XML and RDF, and is being tailored for semantic descriptions and semantic processing. Future work is therefore expected to use semantically richer alternatives (which could be RDF/RDFS, DAML+OIL, or possibly an as-yet-unknown third alternative).

5. Summary

This paper has focused on a prototype editor for aiding the markup of text documents by humans. The markup is currently based on the ability to define regular expressions that can recognize fragments of text. The technique is expected to be suitable for adding markup *post facto* to documents that contain repetitive constructs of text, such as many government documents. The editor is currently designed for use with XML schemas, and extension to other markup formalisms such as RDF and DAML is being investigated. Other means than regular expressions of describing recognition hints are also being investigated.

6. Acknowledgements

This work was supported by National Science Foundation grant EIA-9983267, Sun Microsystems, and the U. S. Coast Guard. The Coast Pilot Division of NOAA (National Oceanic and Atmospheric Administration) supplied the sample chapters of the Coast Pilot. The efforts of Jay Spalding and Oren Stembel in supplying us with the softcopy of the Coast Pilot are gratefully acknowledged. The opinions, findings, conclusions and recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation, NOAA, or the U. S. Coast Guard.

References

[Grosso et al., 1999] Grosso, W. E., Eriksson, H., Fergerson, R. W., Gennari, J. H., Tu, S. W., and Musen, M. A. (1999). Knowledge modeling at the millennium (the design and evolution of Protege-2000). Technical report, Stanford University, Institute for Medical informatics, Stanford, CA. Technical Report SMI-1999-0801.

[Horrocks et al., 2001] Horrocks, I., van Harmelen, F., Patel-Schneider, P., Berners-Lee, T., Brickley, D., Connolly, D., Dean, M., Decker, S., Fensel, D., Hayes, P., Heflin, J., Hendler, J., Lassila, O., McGuinness, D., and Stein, L. A. (2001). DAML+OIL. www.daml.org/2001/03/daml+oil-index.html.

[Malyankar, 2001] Malyankar, R. M. (2001). Maritime information markup and use in passage planning. In *Proceedings of the National Conference on Digital Government*, pages 25–32, Marina del Rey, California. USC/ISI Digital Government Research Center.

[Malyankar, 2002] Malyankar, R. M. (2002). Vocabulary development for markup languages - a case study with maritime information. In *Proceedings of the Eleventh Conference on the World Wide Web, Hawaii*. ACM. To appear.

[Noy et al., 2000] Noy, N. F., Fergerson, R. W., and Musen, M. A. (2000). The knowledge model of Protégé-2000: Combining interoperability and flexibility. In *Knowledge Engineering and Knowledge Management: Methods, Models and Tools: Proceedings of the 12th International Conference, EKAW 2000, Juan-les-Pins, France*, New York. Springer Verlag.

[Spalding and Pirzada, 2001] Spalding, J. and Pirzada, A. (2001). Providing marine navigation information in the new millennium. IAIN World Congress, San Diego.

[Spalding et al., 2002] Spalding, J., Shea, K., Lewandowski, M., and FitzPatrick, M. (2002). Intelligent waterway system and the waterway information network. Institute of Navigation National Technical Meeting, San Diego. (To appear.).

[W3C, 1999] W3C (1999). Resource Description Framework (RDF) model and syntax specification: W3C recommendation 22 February 1999. http://www.w3.org/TR/REC-rdf-syntax/.

[W3C, 2000] W3C (2000). Extensible markup language (XML) 1.0 (second edition). http://www.w3c.org/TR/2000/REC-xml-20001006.

[W3C, 2001a] W3C (2001a). XML schema part 1: Structures. http://www.w3c.org/TR/xmlschema-1/.

[W3C, 2001b] W3C (2001b). XML schema part 2: Datatypes. http://www.w3c.org/TR/xmlschema-2/.