Representing nested semantic information in a linear string of text using XML

Michael Krauthammer, MD¹, Stephen B. Johnson, PhD, George Hripcsak, MD MS¹, David A. Campbell¹ and Carol Friedman¹,²
¹Department of Medical Informatics, Columbia University, New York, ²Queens College CUNY, New York

XML has been widely adopted as an important data interchange language. The structure of XML enables sharing of data elements with variable degrees of nesting as long as the elements are grouped in a strict tree-like fashion. This requirement potentially restricts the usefulness of XML for marking up written text, which often includes features that do not properly nest within other features. We encountered this problem while marking up medical text with structured semantic information from a Natural Language Processor. Traditional approaches to this problem separate the structured information from the actual text mark up. This paper introduces an alternative solution, which tightly integrates the semantic structure with the text. The resulting XML markup preserves the linearity of the medical texts and can therefore be easily expanded with additional types of information.

INTRODUCTION
Extensible Markup Language (XML) (1) is a subset of Standard Generalized Markup Language (SGML) (2) geared for data exchange and processing over the Web. XML is strongly supported by the industry and data standards such as HL7, which has adopted XML for its upcoming RIM Version 3 (3). Recently there has been increasing interest in XML databases and the ability of XML to store semi-structured data (4, 5). Semi-structured data is data that is self-describing and can be processed and stored without an explicit data schema. We are collecting such kinds of data for our data mining activities while parsing free medical text reports with various types of parsers and taggers. The parsing results in different kinds of text annotations such as structured semantic information for diseases and their body location and tagging may identify syntactic part of speech information. As each data mining project 'produces' its own kind of text annotations, we would prefer to store the cumulative annotations of a medical report in a single XML file. Schematically, we would like to be able to mark up a sample sentence such as

the heart is enlarged.

with different kinds of annotation tags as seen in Fig. 1, which shows a sequential syntactic and semantic markup, respectively. Attributes of each element record either part-of-speech information, such as NN (short for noun) or the type of finding, such as cardiomegaly. Ideally, we would envision a growing number of tags added to the existing markup as new research projects produce different kinds of annotations. For example, a future project may be concerned with marking up sensitive data (Fig.1). The growing string of XML tags would therefore be a valuable resource for researchers who want to take advantage of previous analysis of the medical reports.

As XML requires documents to conform to a strict structural composition, in particular in regard to how XML elements nest within other elements, this cumulative storing of annotation may potentially be difficult to achieve. The linearity of the written report may enforce an overlapping markup of the text, which is problematic in XML. To give an example, consider marking up the sentence the auscultation revealed no murmur or gallop.

Semantically, we could mark up the three main concepts featured in the sentence as follows:

the <pr>auscultation</pr> revealed no <f>murmur</f> or <f>gallop</f>
Here, the elements \textit{pr} and \textit{f} stand for procedure and finding, respectively. Expanding the semantic markup to include negation, the markup is not problematic if we consider the negation for the concept \textit{murmur} only:

\texttt{the <pr>auscultation</pr> revealed <f><n>no</n> murmur</f> or <f>gallop</f>.

Here, the element \textit{n} stands for \textit{negation}. Given the meaning of the sentence, the concept \textit{gallop} should also be negated, which results in the following markup:

\texttt{ [...]<f><n>no</n> murmur</f> or gallop</f>.

In this markup, the two \textit{f} elements are overlapping, yielding an incorrect result where the finding \textit{no murmur} is nested in the finding \textit{no gallop} (see Fig. 2a). There are two problems with this representation: First, the nesting implies an invalid \textit{part-of} relation between the two findings, and second, querying the XML tree becomes more difficult, as not all findings are situated on the same tree level.

Table 1 TEI strategies for overlapping text markup

| 1. Boundary marking with milestone elements |
| 2. Reconstitution of virtual element |
| 3. Fragmentation of elements |
| 4. Multiple encoding of the same information |

These strategies are relevant to our problem.

Milestone elements are empty elements, which signify the start and end position of a markup. Because they are empty (i.e. have no content), they do not cause nesting problems in case of overlapping markup. Applied to the sentence above, we get

\texttt{ [...]<fs><fe><n>no</n> murmur</fe> or gallop</fe>.

Here the elements \textit{fs} and \textit{fe} signify the start and end position of the annotation, which results in a flat markup (see Fig. 2b). While an XML parser would be able to parse the above markup, the actual annotation would not be recognized as a nested tree. A postprocessing step would actually be necessary to convert the milestone representation into a regular tree structure.

Reconstitution of virtual elements is a commonly used strategy to deal with overlapping markup. At our institution, we use a natural language processor called MedLEE (7-9), which transforms different kinds of medical text reports (x-ray reports, discharge summaries) into structured XML (10). The representation creates numbered text elements, which are linked to semantic elements (virtual

![Fig. 2. XML tree structures (I)](image)

Fig. 2. XML tree structures (I) elements) in a separate part of the document. This is shown schematically in Fig. 3. The semantic information constitutes the procedure \textit{auscultation}, and the findings \textit{no murmur} and \textit{no gallop}, which are linked to numbered words in the text. This linking can be represented in XML using identity attributes.

In case of MedLEE, the text elements are called \textit{phr} (for phrase) and are located in the text-section of the document (Fig. 4). The \textit{phr} elements are linked via the \textit{id/idref} attribute (usually unique identifiers rather than numbers as shown here for illustration purposes) to the semantic findings in the structured section of the document.

![Fig. 3. Semantic information linked to words in the text](image)

Fig. 3 semantic information linked to words in the text

As can be easily seen, this representation correctly handles the negation of the elements \textit{murmur} and \textit{gallop}. As with the milestone elements, a postprocessor is needed to transform this split representation into a nested tree, which includes semantic and textual information together.

![Fig. 4. MedLEE format](image)

Fig. 4. MedLEE format

\texttt{<structured>}
<procedure v="auscultation" idref="2" />
<finding v="murmur" idref="5" certaintv="no"
 idref="4"/></finding>
<finding v="gallop" idref="7" certaintv="no"
 idref="4"/></finding>
</structured>

\texttt{<text>the <phr id="2">auscultation</phr> revealed
<phr id="4">no</phr> <phr id="5">murmur</phr> or
<phr id="7">gallop</phr>.</text>
Two further TEI strategies, fragmentation of existing elements as well as multiple encoding of the same information, are not really suited for our problem because they either do not code all necessary elements or increase the complexity of the markup. Beyond TEI, there are other solutions dealing with concurrent markup. For example, “standoff annotation” introduces two separate documents for the markup and the actual text corpus (11). Other solutions use XPath expressions (12) to encode overlapping hierarchies (13), or use advanced markup grammars supporting concurrent markup (14).

We propose an alternative representation, which combines ideas from both the milestone and virtual elements strategy. Our main goal is to keep the text and annotation together while minimizing the need for empty elements and link attributes. In this representation, empty elements are not used as starting and end points of a markup, but rather as semantic elements, which link to remote text elements. The resulting representation should be linear and maintain the word order of the original text, thus be suitable for inclusion into a semi-structured database. The linear structure guarantees the ability to add new annotation where necessary as well as fast query performance through maintaining a single main XML tree.

METHODS AND RESULTS
The problem can be generalized as follows: Given a string of words and a set of annotations which refer to any combination of those words in the string, assign annotations with no overlap and with minimal empty elements and link attributes. Given our example sentence above, a possible markup which satisfy those requirements would look as follows:

the <pr>auscultation</pr> revealed <f>n id="4"/> no</n> murmur</f> or <g>gallop</g>n idref="4"/>

This representation (which we will call linearized during the remainder of this paper) includes a single link (the negation of the finding gallop is linked to the actual word no in the text) as well as a single empty element (the negation of the finding gallop). This markup produces a properly nested structure as can be seen in Fig 5a.

Given the current MedLEE XML markup format discussed above, which uses virtual elements linked to numbered text elements, we wanted to explore whether we could generate the linearized representation automatically. We developed an algorithm suited for this task. The main idea is to represent each semantic concept (finding, procedure etc.) as an ordered set consisting of the element numbers of the text. The algorithm decides which text elements will be included in the final markup of each concept.

**Fig. 5. XML tree structures (II)**

![XML tree structures (II)](image)

By marking up one semantic concept after another, the sets dynamically change their content. Given the above example sentence, the sets would initially look as follows (see also Fig.3):

<table>
<thead>
<tr>
<th>Semantic concept</th>
<th>Sets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Auscultation</td>
<td>(2)</td>
</tr>
<tr>
<td>No murmur</td>
<td>(4,5)</td>
</tr>
<tr>
<td>No Gallop</td>
<td>(4,7)</td>
</tr>
</tbody>
</table>

The sets corresponds to the numbers of the text elements in the text:

the &lt;2&gt;auscultation&lt;/2&gt; revealed &lt;4&gt;no&lt;/4&gt; &lt;5&gt;murmur&lt;/5&gt; or &lt;7&gt;gallop&lt;/7&gt;.

The algorithm sorts the sets by listing the set containing the lowest element numbers on top. During the first iteration, the algorithm would then mark up the semantic concept corresponding to the set on top - in this case auscultation. The algorithm decides which text elements in the set are included in the markup. The rule is as follows: Include every element whose number is equal or smaller than the element numbers of the next concept - in this case no murmur. As 2 is smaller than any number in the next set, the text element number 2 is included in the markup of auscultation.

the &lt;pr&gt;auscultation&lt;/pr&gt; revealed &lt;4&gt;no&lt;/4&gt; &lt;5&gt;murmur&lt;/5&gt; or &lt;7&gt;gallop&lt;/7&gt;.

The number 2 is removed from the sets, and the remaining sets are again sorted. Concept no murmur is now on top, corresponding to a set with text element numbers 4 and 5.

<table>
<thead>
<tr>
<th>Semantic concept</th>
<th>Sets</th>
</tr>
</thead>
<tbody>
<tr>
<td>No murmur</td>
<td>(4,5)</td>
</tr>
<tr>
<td>No Gallop</td>
<td>(4,7)</td>
</tr>
</tbody>
</table>

The first number, 4, is equal to the first number of the set corresponding to no gallop and included in the markup.
At this point, the table stores the number 4 as being included (no murmur) as well as excluded (no gallop) in the text markup. The two remaining sets contain a single element number, and the algorithm assigns number 5 to the concept on top — no murmur. Marking up the latter concept demands a link attribute at position 4, which corresponds to the word no, and serves as a reference for concept no gallop.

The markup of the last concept — no gallop - is a straightforward process.

The fact that text element 4 was excluded from the markup necessitates an empty ‘negation’ element with a link attribute. The resulting markup looks as follows (see also Fig. 5a):

the `<pr>ausculation</pr> revealed `<f><n id="4">no</n><f><bl/`murmur<bl/> or `<f><bl/`gallop<bl/>.

Looking at a different example, it turns out that seemingly short sentences may contain surprisingly many semantic concepts. Consider a sentence

*erosions and bleeding in the antrum and fundus.*

MedLEE recognizes 4 different medical concepts in this sentence: *erosion antrum, erosion fundus, bleeding antrum and bleeding fundus.* Marking up such a sentence is problematic. For example, the finding *bleeding* separates the finding *erosion* from the body location *antrum*, causing potential nesting problems.

The numbered sentence looks as follows:

`<f>erosions</f>` and `<f>bleeding</f>` in the `<f>antrum</f>` and `<f>fundus</f>`.

The table of sets reveals a more complex situation than above.

<table>
<thead>
<tr>
<th>Semantic concept</th>
<th>Sets</th>
<th>Included</th>
<th>Not included</th>
</tr>
</thead>
<tbody>
<tr>
<td>Erosion antrum</td>
<td>(1,6)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Erosion fundus</td>
<td>(1,8)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bleeding antrum</td>
<td>(3,6)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bleeding fundus</td>
<td>(3,8)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In this case, the algorithm would perform the same iterations as before, backtracking after step 2 to resolve a situation where two concepts correspond to the same set of text elements. The final markup would look as follows (ids/idrefs not shown, see also Fig. 5b):

`<f>erosions</f>` and `<f>bleeding</f>` in the `<f>antrum</f>` and `<f>fundus</f>`.

The element bl stands for body location, such as *antrum or fundus.* As can been seen, although there was considerable overlap in the original semantic markup, obtaining a linearized representation is feasible.

This algorithm has been run on more complex sentences (see Fig. 6), as well as on a test set of over 3000 discharge summaries, which have been parsed by MedLEE. A first analysis shows that the algorithm successfully linearized over 90’000 sentences. A validation of the algorithm can be achieved by first automatically linearizing the original split MedLEE format (see above), and subsequently transforming the linearized representation back to the split format. If we can reconstruct the original format, we could demonstrate that the algorithm is working and conserving all data.

**DISCUSSION AND CONCLUSION**

There is growing interest in so-called semi-structured XML databases (4, 5), which are able to store data without an explicit data schema. This is convenient for our data mining activities involving medical text reports, where different projects generate various kinds of text annotations, which we would like to store in a single XML file. As some of these annotations will be overlapping, it is difficult to generate a file with a valid XML structure. By marking up medical texts with semantic information from Natural Language Processing (NLP), we observed this problem especially with conjunction *(and)* or disjunctions *(or)* in texts. We use such conjunctions and disjunctions in our daily language to conveniently communicate several facts in the shortest form possible. Consider a sentence

No bleeding in the antrum and fundus.

*Expanding* the above sentence reveals two different facts, [no bleeding antrum] and [no bleeding fundus]. Annotating such overlapping information with XML is difficult without generating an invalid nesting of elements. Traditional approaches to this problem either separate the annotation from the text by creating so-called virtual elements or use empty elements to indicate the start and end positions of a specific annotation. Both approaches have their disadvantages: virtual elements separate the documents in two parts (text and annotation), and empty elements must be converted to regular elements in order to represent a nested tree structure. The linearized representation of XML presented in this paper keeps the XML tags and the text together while minimally relying on link attributes as well as empty elements. A possible advantage of this markup strategy is therefore reduced storage requirement. XML is known to increase storage requirement
considerably by creating structural overhead. By reducing the number of elements and attributes, the linearized markup may turn out to be storage efficient. Another advantage of linearization may be improved query performance across documents. The linearized markup maintains a single XML tree, and may thus show better query performance when retrieving text and annotation together. In summary, we think this representation is a valid alternative to other markup solutions.

This work was supported by National Library of Medicine grants R01-LM06910 and R01-LM06274.

Fig. 6. Sentence distributing two elements, bodyloc (bodylocation) and certainty, across three findings, clubbing, cyanosis and edema. The negation is encoded as an attribute in the certainty element.

Sentence: 'Vascular: no clubbing, cyanosis, or edema.'

Findings (as identified by MedLEE):

1. No clubbing, vascular
2. No cyanosis, vascular
3. No edema, vascular

Linearized markup:

- <finding v="clubbing">
  - <bodyloc v="vessel" id="1">
    Vascular
    </bodyloc>
  - <certainty v="no" id ="2">
    no
    </certainty>
  clubbing
  </finding>

- <finding v="cyanosis">
  cyanosis
  <bodyloc v="vessel" idref="1" />
  <certainty v="no" idref="2" />
  </finding>

, or

- <finding v="edema">
  edema
  <bodyloc v="vessel" idref="1" />
  <certainty v="no" idref="2" />
  </finding>

REFERENCES