'Help, my XML is too complex!' – the problem of excessive structural markup in dictionaries

Michal Měchura

Natural Language Processing Centre, Masaryk University, Brno, Czech Republic

Abstract

Dictionary encoding is the activity of taking an inventory of lexicographic object types such as headword, part-of-speech label, sense and translation, and expressing them formally in a data serialization language such as XML. But the use of XML for dictionary encoding often leads to excessively complex markup, with multi-layered embedding of elements inside other elements inside yet more elements. The following code shows how a pair of translations would typically be encoded in a bilingual dictionary (adapted from the New English–Irish Dictionary [Ó Mianáin and Convery 2014]).

Code sample 1

```
<translations>
  <translationContainer>
    <translation>leasú</translation>
    <pos>n-masc</pos>
  </translationContainer>
  <translationContainer>
    <translation>athchóiriú</translation>
    <pos>n-masc</pos>
    <usage>formal</usage>
  </translationContainer>
</translations>
```

The only XML elements here that contain actual human-readable information are <translation> (the translation's wording), <pos> (its part of speech) and <usage> (its usage label). The remaining XML elements are purely structural, used for grouping other elements together. Arguably, their presence here distracts a human XML reader (and even more so, a human XML writer) from lexicographic information which is otherwise simple and could be expressed more economically in some other (not yet existent) serialization language such as the pseudo-code in the following code sample.

Code sample 2

```
translation: leasú
  pos: n-masc
translation: athchóiriú
  pos: n-masc
  usage: formal
```

The distracting presence of purely structural elements in lexicographic XML is often acknowledged as an inconvenience in e-lexicographic circles informally but, to the author's knowledge, no serious attempts have been made yet to analyze or solve it.

We can define *purely structural markup* as such XML elements which contain no text nodes as their direct children: all their child nodes are other XML elements. Purely structural elements tend to be called *groups, containers* or *blocks* in the entry schemas of various dictionaries. For example, the entry schema for the DANTE project (Atkins, Kilgarriff and Rundell 2010) consists of elements such as <CollocGp> (collocate group) as a wrapper for a sequence of one or more collocates, <CollocCont> (collocate container) as a wrapper for a single collocate along with additional information about it (usage labels, example sentences, translations etc.) and finally <COLLOC> as a wrapper for the actual collocate (a text node). The first two of these three element types are purely structural. Broadly speaking, we tend to find two patterns of purely structural markup in lexicographic XML.

List pattern. The first kind is a pattern in which a parent element wraps a sequence of child elements which are all of the same type, such as <CollocGp> for a series of collocates in Dante, or <translations> for a series of translations in Code sample 1. They are almost always unnecessary in the sense that they

convey no useful information. They are there because the designer of the entry schema probably thought it 'logical' to group elements of the same type under a common parent element. But the usefulness of this grouping is debatable: the group thus created does not seem to represent any lexicographic fact which a lexicographer might want to communicate to the dictionary's end-users. Unnecessary grouping of this kind can be found in XML outside lexicography too and tends to be advised against in XML styleguides (eg. Ogbuji 2004).

Headed pattern. The second kind is a pattern in which a parent element wraps child elements of different types, one of which can be considered the head and the others can be seen as providing additional information about the head. An example is <translationContainer> in Code sample 1 which can be said to be headed by <translation>, while the other children <pos> and <usage> provide additional information about the head. In DANTE, a similar example is <CollocCont> which is headed by <COLLOC> (the actual collocate) while other child elements of <CollocCont> provide additional information about the head (usage labels, example sentences, translations etc.). Unlike the list pattern, the headed pattern cannot be explained away as a bad practice. Its purpose is to encode a lexicographic fact which the lexicographer wants to communicate to the end-user: for example, which <pos> element modifies which <translation> element. The purely structural <translationContainer> element is a tool for encoding that fact.

In this paper I will focus on the headed pattern of purely structural markup and dicuss it in depth. I will identify several subtypes of this pattern and show examples from real-life dictionaries. I will discuss whether it is possible to encode the headed pattern in XML *without* recourse to purely structural markup (for example by using XML attributes), but I will reach the conclusion that some amount of purely structural markup is unavoidable and that the problem is unsolvable, as long as one insists on using XML.

Secondly, I will evaluate other popular data serialization languages such as JSON a YAML and show that they, too, lead to excessive structural markup. What XML, JSON and YAML have in common is that they take no account of the inherent *headedness* of many lexicographic information objects such as collocations, example sentences and translations. In XML and other languages, the only way to encode the relation between a head (such as <translation>) and its modifiers (such as <pos> or <usage>) is to wrap them inside a common parent (such as <translationContainer>), which unavoidably leads to the proliferation of excessive structural markup we see in lexicographic XML everywhere.

An ideal lexicographic serialization language would be one which respects the inherent headedness of lexicographic data. In conclusion I will propose the creation of such a *lexicographic lightweight markup language* (à la Benko 2018). The language would read similarly to the pseudo-code in Code sample 2 and could be used in dictionary writing systems either as a replacement for XML or as a non-persisten surface representation on top of XML in the fashion of *Invisible XML* (Pemberton 2013).

Keywords: XML, dictionary encoding, dictionary editing, lexicographic lightweight markup language

References

Atkins, B. T. S.; Kilgarriff, A.; Rundell, M (2010). Database of ANalysed Texts of English (DANTE): the NEID database project. In: Proceedings of the Fourteenth EURALEX International Congress, EURALEX 2010.

Benko, V. (2018). In Praise of Simplicity: Lexicographic Lightweight Markup Language. In: Proceedings of the XVIII EURALEX International Congress: Lexicography in Global Contexts.

Ogbuji, U. (2004). Considering container elements: When to use elements to wrap structures of other elements. In: Principles of XML design, IBM, https://www.ibm.com/developerworks/library/x-contain/index.html (accessed 6 September 2019).

Ó Mianáin, P.; Convery, C. (2014). From DANTE to Dictionary: The New English-Irish Dictionary. In: Proceedings of the Sixteenth EURALEX International Congress, EURALEX 2014.

Pemberton, S. (2013). Invisible XML. In: Proceedings of Balisage: The Markup Conference 2013. Balisage Series on Markup Technologies, vol. 10.