Shareable subentries in Lexonomy as a solution to the problem of multiword item placement

Michal Boleslav Měchura
Masaryk University, Brno, Czech Republic
michmech@mail.muni.cz

Abstract

This paper introduces a new way of dealing with phraseology in dictionaries. A classical question in lexicography is whether multiword items such as *third time lucky* should be listed under *third*, *time* or *lucky*. The ideal answer is ‘under all of them’ but, until now, the only way to do that in conventional tree-structured dictionaries has been to keep multiple copies (of what conceptually is one and the same item) in several places throughout the dictionary. We present a way to achieve the same goal without copying. The multiword item becomes a semi-independent subentry which exists in only one copy but appears simultaneously in several places in the dictionary. The structure of the dictionary remains a tree but the lexicographer is empowered to occasionally ‘break out’ of the tree in order to avoid duplication. This paper explains the reasoning behind the concept of shareable subentries and shows how this new functionality has been implemented in the dictionary writing system Lexonomy.

Keywords: subentries; phraseology; Lexonomy

1 The problem of multiword item placement

A perennial problem in lexicography is deciding on the placement of multi-word items (Bogaards 1990): should a phraseological unit such as *third time lucky* be located inside the entry for *third, time* or *lucky?* In many such cases the best imaginable answer is 'under all of them'. But such a suggestion is difficult to accommodate in the classical model of dictionary entries as a tree structures. The only way to include a phraseological unit in more than one entry is to duplicate it, but this is an inelegant solution. Most importantly, it opens up the potential for inconsistency: if a lexicographer makes a change to the subentry *third time lucky* under *third*, there is no automatic way to propagate the change to the other copies under *time* and *lucky.*

A popular method to deal with this in born-digital dictionaries is to treat multi-word phrasemes as independent entries, in effect promoting them to the same level as single-word headwords. This approach ‘solves’ the problem of multiword item placement by deciding not to place them anywhere, and that is also its drawback: it strips the lexicographer of the ability to include a multiword item like *third time lucky* in a specific sense of a single-word entry, for example in a specific sense of *time*. Instead, it delegates the placement question to the search algorithm, hoping that *third time lucky* will indeed appear somewhere on the user’s screen when the user has looked up *time*. This is far from ideal: the job which an item like *third time lucky* does in a dictionary is not just that of a phraseme which users might look up independently. It is (or can be) simultaneously an illustrative example of specific senses of the words it is composed of. This means that the desire to include it in a specific location inside one or more specific entries is lexicographically well-motivated and the ‘treat-multiwords-as-headwords’ method is only a workaround. What is needed is a method for including a single multiword item in several locations inside several entries, but without having to keep multiple copies of them in multiple locations.

2 From trees to graphs (and then back a little)

The almost total computerization of lexicography in recent decades has not solved the multiword item placement problem. Dictionary entries are usually encoded as XML documents, a formalism which, while making the structure of an entry explicit, offers no innovative departures from the classical tree-structured model.[1] In XML, if one wishes to share an XML fragment between several XML documents, one has to resort to extensions such as XLink which require additional processing and which lack implementation in existing dictionary writing systems and other XML tools.

This limitation of the classical tree-based model has inspired some authors to re-imagine dictionaries as graphs rather than trees. In a graph, an element (for example, a phraseological subentry) can be connected to any number of other elements (for example, to several senses of several headwords). Compare this to tree-structured XML where every element can only be contained inside one other element (for example, a phraseological subentry can only be contained inside one sense of one headword).

Curiously, graph-based dictionaries have not become the norm in (human-oriented) lexicography. While graph-structured datasets are common in machine-oriented lexicons (e.g. various wordnets), human-oriented lexicography has

[1] It would be tempting to assume that the modelling of dictionary entries as tree structures was an innovation introduced into lexicography by the arrival of XML. That is not true. The ‘imagining’ of dictionary entries as theoretical tree structures predates even the invention of XML. A summary of this thinking from pre-computarization times can be found in Wiegand (1989). I am grateful to David Lindemann for this insight.

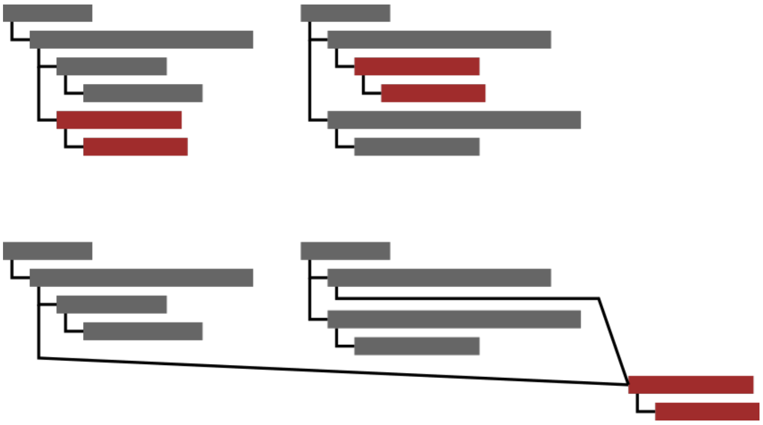
been reluctant to adopt the graph formalism. Existing implementations are rare (e.g. Polguère 2004) and the graph often serves only as an export format for what was originally a tree-structured dictionary: this is the case for most human-oriented lexicons on the Semantic Web (e.g. Aguado-de-Cea et al. 2016, Klimek and Brümmer 2015) where the dictionaries are exported from proprietary tree-structured XML into RDF graphs. Most of mainstream lexicography has remained firmly committed to the tree paradigm. One consequence is that the problem of multi-word item placement remains unsolved.

The disadvantage of graphs (such as RDF graphs on the Semantic Web) is that they are not as easily human-readable as XML trees (and trees in general), not to mention human-writeable. Trees can be visualized neatly as two-dimensional objects, while graphs often can't. Trees are easy for humans to grasp mentally, while graphs are more difficult to 'take in'. For this reason, it is unlikely that lexicographers will switch to authoring graph-based dictionaries directly any time soon.

The problem then is that, while graphs are the more adequate structure for dictionaries, trees are more 'lexicographer-friendly'. What we need is a compromise: a set up which keeps dictionaries in a tree-like structure as much as possible, but which also allows them to 'break out' of the tree when necessary: for example to allow the sharing of phraseological subentries between entries (Figure 1). Importantly, we also need a dictionary writing system which allows lexicographers to work with dictionary entries in the familiar tree format as much as possible, while only forcing them to 'think outside the tree' when necessary.

In this paper we present how the dictionary writing system Lexonomy[2] (Měchura 2017) offers such a compromise by introducing the concept of **shareable subentries.** A dictionary administrator, while defining the entry schema, can designate certain sections of the XML tree, for example phraseological subentries, to be 'shareable', turning them into snippets of XML which are allowed to appear in several entries simultaneously. The structure of the dictionary remains a tree but the mechanism of shareable fragments empowers the lexicographers to 'break out of the tree' in order to avoid duplication. This structure is a compromise between trees and graphs and could perhaps be described as 'graph-augmented trees' (Měchura 2016).

Figure 1: In a classical tree-structured dictionary (above) each element can have only one parent, leading to unnecessary duplication of data across entries. In the graph-augmented tree structure proposed in this paper (below) an element can have multiple parents, allowing reuse of the same element in multiple entries.



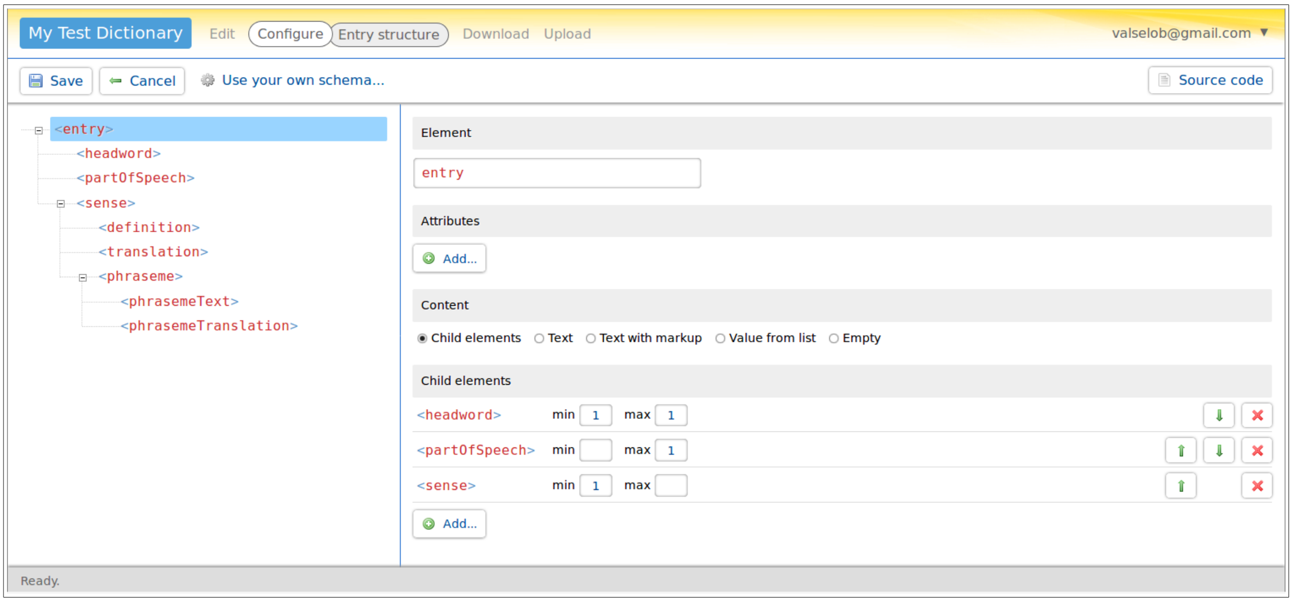
3 Working with shareable subentries in Lexonomy

Setting up a dictionary with subentries in Lexonomy begins just like setting up any dictionary in Lexonomy: you need to create the entry structure first (Figure 2). This is where you decide which XML elements your entries will be made up of, what their names will be, how they will stack up inside each other and so on. In this example we have a very simple bilingual dictionary where each entry has a headword, an optional part-of-speech label and one or more senses. Each sense can have translations and something called *phrasemes:* these will be our multiword items such *third time lucky.*

The next step is to tell Lexonomy that we want phrasemes to be shareable subentries. We do this in the *Subentries* section of the dictionary's configuration screen (Figure 3). All elements we list here will be treated as subentries by Lexonomy and we will be able to share them among several entries.

[2] https://www.lexonomy.eu/

Figure 2: Setting up the entry structure of a new dictionary.



```
<entry>
  <headword>
  <partOfSpeech>
  <sense>
    <definition>
    <translation>
    <phraseme>
      <phrasemeText>
      <phrasemeTranslation>
```

Element: entry

Attributes: Add...

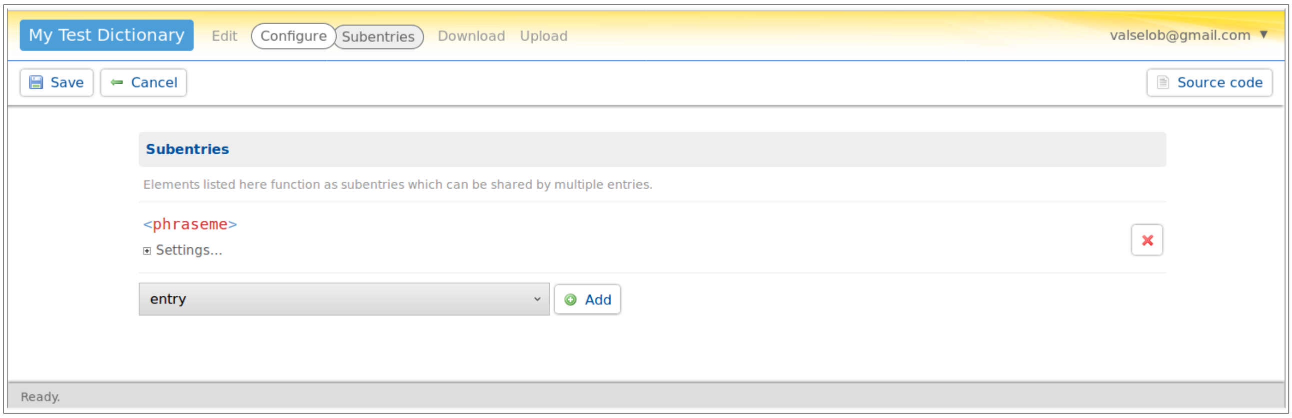
Content: ● Child elements ○ Text ○ Text with markup ○ Value from list ○ Empty

Child elements

<headword>	min	1	max	1
<partOfSpeech>	min		max	1
<sense>	min	1	max	

Add...

Figure 3: Telling Lexonomy which XML elements should be treated as shareable subentries.



Subentries

Elements listed here function as subentries which can be shared by multiple entries.

<phraseme>

⊞ Settings...

entry ⌄ Add

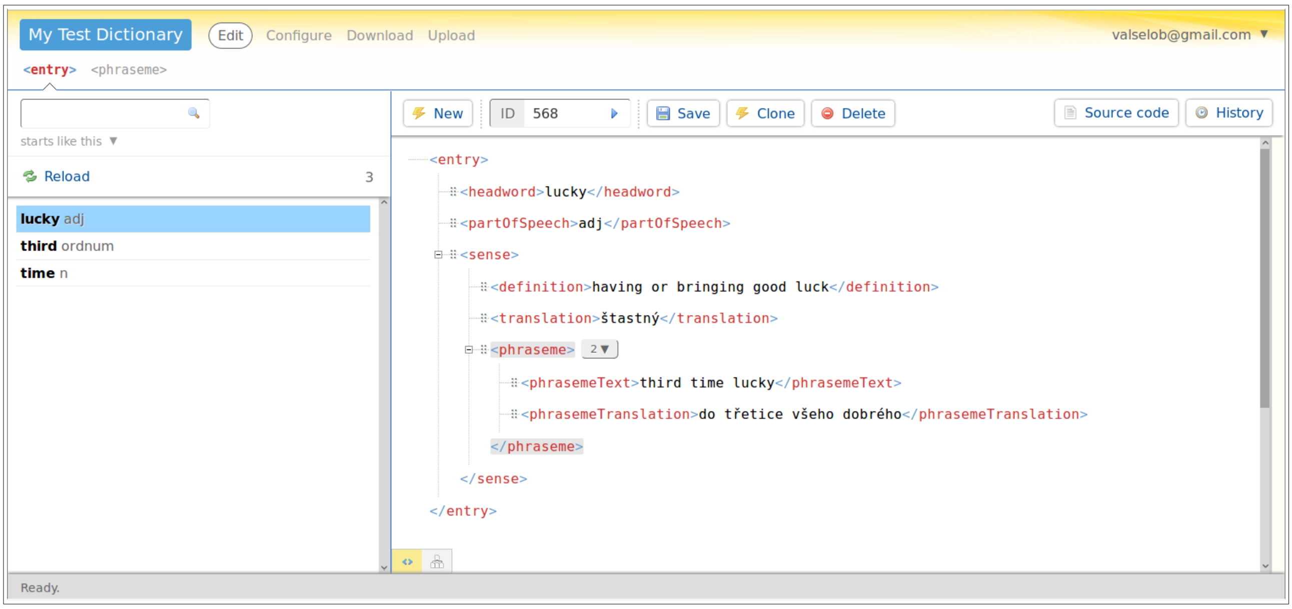
Now that the dictionary has been configured we can have a look at how lexicographers work with entries and subentries in the editing interface. Figure 4 shows the entry for *lucky* being edited. You will probably agree that this looks almost exactly like entries usually look in Lexonomy, with the XML source visible and open for editing. The only difference is that the <phraseme> element has a shaded background: this is Lexonomy's way of telling us that this element (with all its children) is a shareable subentry and may be shared with other entries. The button near the element tells us how many other entries, besides this one, share this subentry: in this case, two. You can click the button to see which entries those are (Figure 5).

This is the point at which lexicographers need to mentally 'break out' of the tree and realize that any changes made to the content of <phraseme> here will automatically be propagated into the other two entries that share this phraseme. In effect, you are editing not just the entry you are looking at now but the other ones too.

Now let's have a look at how subentries are added into entries. As you probably know, every XML element in Lexonomy's editing interface has a menu which you can open by clicking the element's name. To add a new <phraseme> to a <sense> click the <sense> and a menu will appear. Because Lexonomy knows that <phraseme> elements are shareable, one of the options it will offer you is an option to find <phraseme> subentries that already exist elsewhere in your dictionary (Figure 6). Clicking it will bring up a window where you can search all the <phraseme> elements that already exist anywhere in your dictionary (Figure 7). You can tick the ones you want added to the sense and click the *Insert* button. If you have not found a suitable phraseme that exists already, create a new one by clicking the *New* button: this will insert empty XML markup into your entry and you can fill it in (Figure 8). Once you have saved the entry this newly created <phraseme> subentry will join the ranks of other subentries in your dictionary and will be available for sharing with other entries.

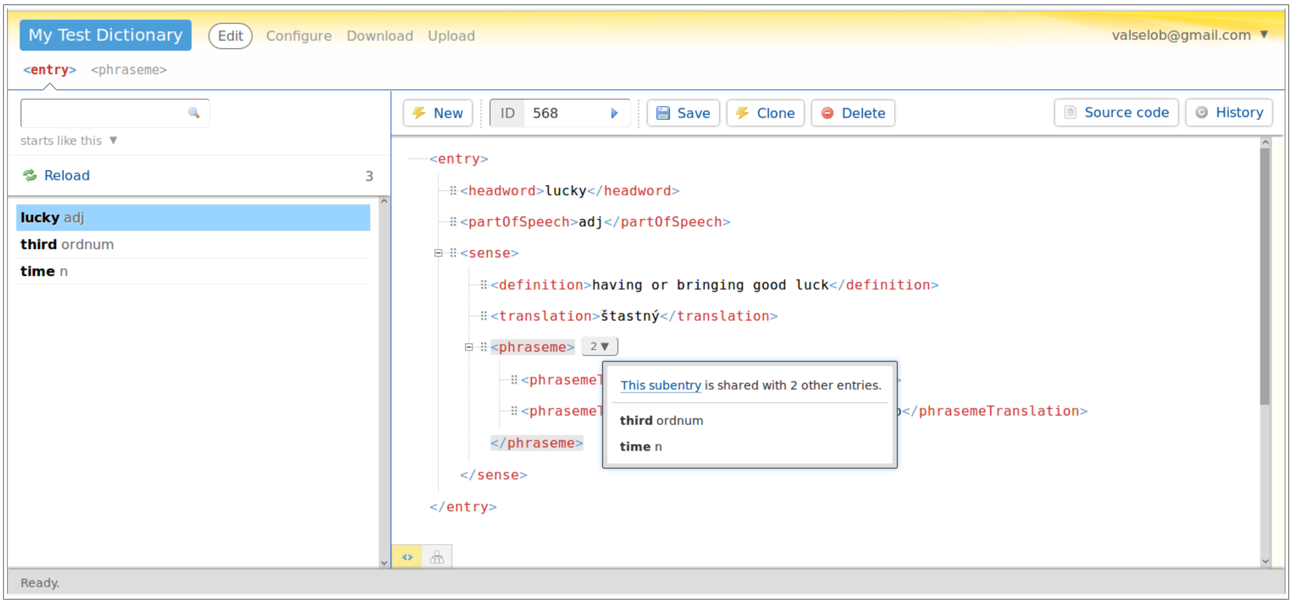
As you populate your dictionary with entries you will gradually accumulate a collection of shareable subentries inside the dictionary. This collection of subentries is almost like a separate sub-database in your dictionary and, if you want, you can look at it in isolation. Notice that, in the top left corner of the screen, Lexonomy gives you the option to choose which type of entries you want to work with: the main entries whose top-level element is <entry> or the subentries whose top-level element is <phraseme> (Figure 9). It goes without saying that any changes you make to a subentry here will be automatically propagated to all the entries that contain it.

Figure 4: Editing an entry which contains a subentry.



```
<entry>
  <headword>lucky</headword>
  <partOfSpeech>adj</partOfSpeech>
  <sense>
    <definition>having or bringing good luck</definition>
    <translation>šťastný</translation>
    <phraseme>
      <phraseText>third time lucky</phraseText>
      <phraseTranslation>do třetice všeho dobrého</phraseTranslation>
    </phraseme>
  </sense>
</entry>
```

Figure 5: Lexonomy tells us which other entries have the same subentry.



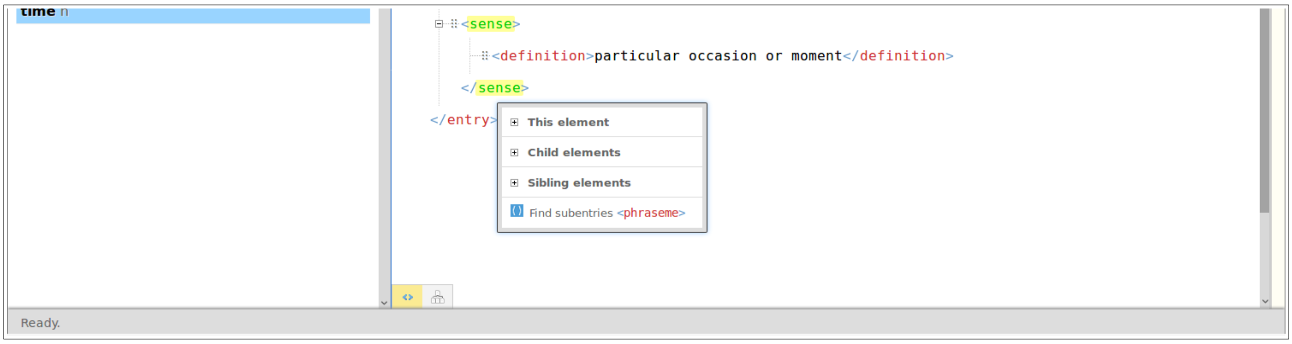
```
<entry>
  <headword>lucky</headword>
  <partOfSpeech>adj</partOfSpeech>
  <sense>
    <definition>having or bringing good luck</definition>
    <translation>šťastný</translation>
    <phraseme>
      <phraseme
      <phraseme                              </phraseTranslation>
    </phraseme>
  </sense>
</entry>
```

This subentry is shared with 2 other entries.

third ordnum

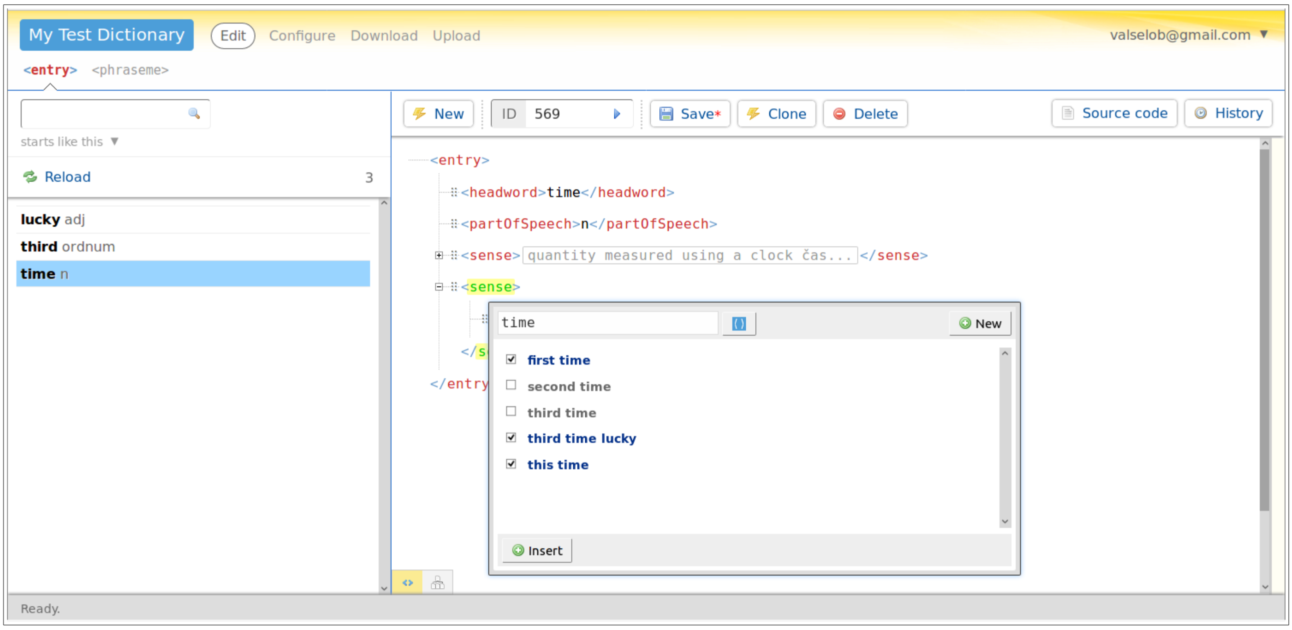
time n

Figure 6: This menu contains an item for finding existing subentries.



```
  <sense>
    <definition>particular occasion or moment</definition>
  </sense>
</entry>
```

Figure 7: Adding subentries into an entry.



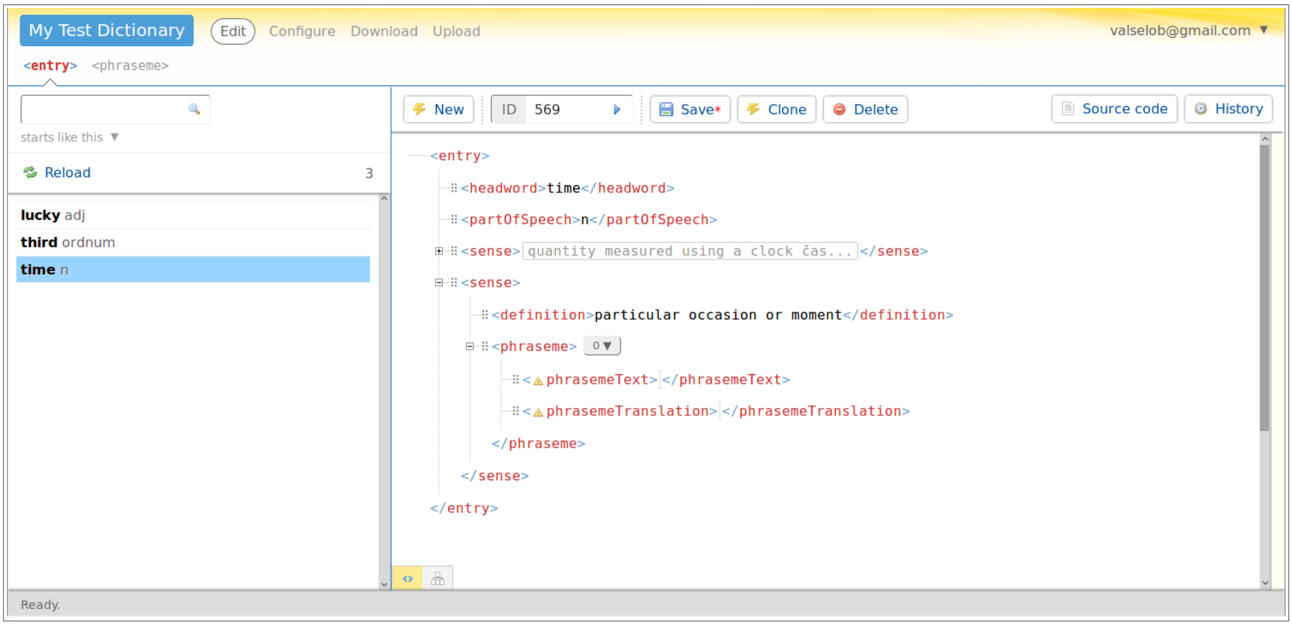
lucky adj
third ordnum
time n

```
<entry>
  <headword>time</headword>
  <partOfSpeech>n</partOfSpeech>
  <sense> quantity measured using a clock čas... </sense>
  <sense>
  </s
</entry
```

time

- ☑ first time
- ☐ second time
- ☐ third time
- ☑ third time lucky
- ☑ this time

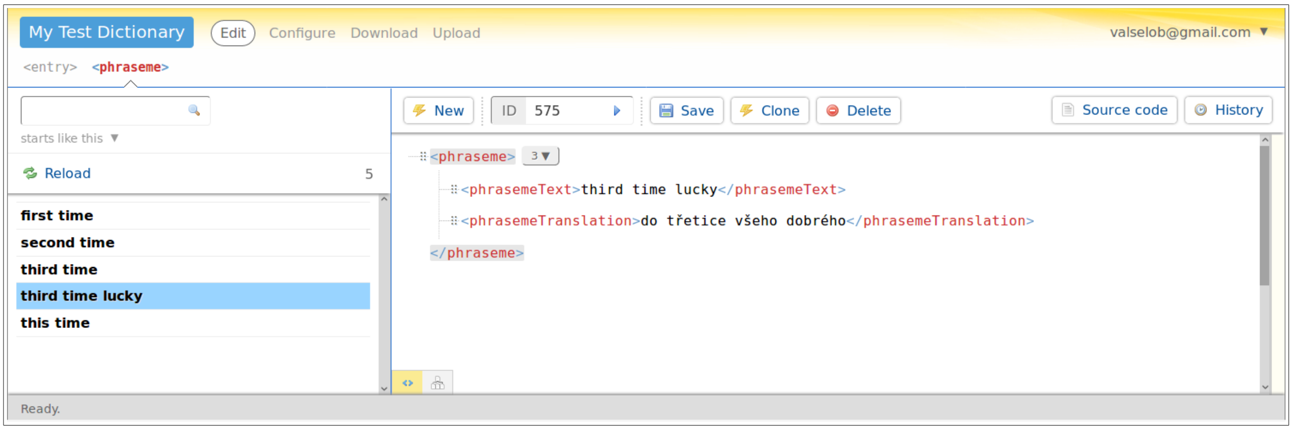
Figure 8: Creating a blank new subentry is the same as inserting a new XML element.



lucky adj
third ordnum
time n

```
<entry>
  <headword>time</headword>
  <partOfSpeech>n</partOfSpeech>
  <sense> quantity measured using a clock čas... </sense>
  <sense>
    <definition>particular occasion or moment</definition>
    <phraseme> 0 ▼
      <phrasemeText></phrasemeText>
      <phrasemeTranslation></phrasemeTranslation>
    </phraseme>
  </sense>
</entry>
```

Figure 9: Looking at subentries in isolation.



first time
second time
third time
third time lucky
this time

```
<phraseme> 3 ▼
  <phrasemeText>third time lucky</phrasemeText>
  <phrasemeTranslation>do třetice všeho dobrého</phrasemeTranslation>
</phraseme>
```

4 Taking the idea further: what can be a subentry?

The original motivation for shareable subentries in Lexonomy was multiword phraseological units. But once such a feature exists, the next obvious question to ask is, what other things can it be used for? And it turns out that it can be used for quite many other things. Everywhere we want to avoid having to duplicate the same information in several places, shareable subentries should be considered.

One obvious candidate is example sentences. A sentence like *That's just great!* would work equally well as an example under *great* and under *just.* So this would be an argument in favour of turning the examples into shareable subentries in your dictionary in Lexonomy – especially if the examples carry some other data than just the wording itself, such as pragmatic labels or translations into another language. Then each example is almost like a mini-entry in its own right and might look something like this:

```
<example>
  <exampleText>That's just great!</exampleText>
  <label>sarcasm</label>
  <exampleTranslation>Einfach großartig!</exampleTranslation>
</example>
```

It would be unwise to duplicate this entire fragment in two separate entries (one copy under *great* and one under *just*) because that would be an invitation to inconsistency. If a lexicographer changes one copy (correcting the translation, let's say, or changing the pragmatic label) there is no guarantee that they will remember to change the other copy as well. A wiser approach is to turn the `<example>` element into a shareable subentry. It makes no difference that most such subentries will in fact not be shared (will only occur in one entry): what is important is that the few that will, will always remain synchronized and consistent. Another advantage is that you can treat your example sentences as a separate sub-database inside the dictionary. You can even prepare a large dataset of example sentences beforehand, upload them into Lexonomy (using the *Upload* feature) and then compose all your dictionaries entries from them as building blocks.

At this point an inquisitive reader might ask whether it is possible to have subentries inside subentries in Lexonomy. The answer is yes. It is possible (and in fact sensible) to have, let's say, both phrasemes and examples configured as shareable subentries. Some examples will occur at sense level (outside any phraseme) and some will be inside phrasemes. When a phraseme is shared among multiple entries, it will take all its examples along everywhere it goes and they will be shared too.

Another candidate for 'shareability' is translation equivalents in bilingual dictionaries. Let's assume you are working on a bilingual encoding dictionary, that is, a dictionary which is meant to help its users produce texts in a language which is not their mother tongue. In such a dictionary the translations of the headwords are likely to contain a lot of data in addition to the word itself, such as grammatical labels, pronunciation transcriptions and inflected forms. Once again, we find ourselves in a situation where a translation is almost its own mini-entry and could look like this:

```
<translation>
  <translationText>Wohnsitz</translationText>
  <pronunciation>ˈvoːnˌzɪʦ</pronunciation>
  <partOfSpeech>noun</partOfSpeech>
  <gender>masc</gender>
  <plural>Wohnsitze</plural>
</translation>
```

A translation like the German *Wohnsitz* is likely to be found under many different English headwords: *residence, domicile, abode...* So it makes sense to turn the `<translation>` element into a shareable subentry and avoid unnecessary duplication. Once a lexicographer has created the `<translation>` element for *Wohnsitz* in one entry, it can be reused in other entries without having to retype all the information, and without risking inconsistency. And again, it might make sense to treat your translation equivalents as a separate sub-database, prepare a long list of them beforehand, upload them into Lexonomy, and then compose your dictionaries from them as building blocks.

For a final and most extreme example, let's consider the idea of turning headwords themselves into shareable subentries. In a decoding (as opposed to encoding) dictionary, it is the headwords (as opposed to the translations) which carry a lot of grammatical and other annotations. Now, headwords are normally not shared among entries (except when homonyms are given separate entries) but it is possible to imagine unconventional dictionaries where they are: for example, a valency dictionary where each valency pattern is treated as a separate entry. Then we would have, for each headword, many entries which share the headword. In that case it would make sense to turn headwords (along with all their grammatical and other annotations) into shareable subentries.

The conclusion is that the mechanism of shareable subentries is suitable for every situation where we are facing the risk of duplication, which is not limited to multiword phraseological subentries.

5 How subentries are implemented in Lexonomy

Lexonomy is open-source software where everybody can see and even adapt the source code. For those who might be interested in doing that, a short explanation is in order of how Lexonomy handles subentries internally. Every XML element which is a shareable subentry has an attribute called `lxnm:subentryID` (`lxnm` refers to the namespace `http://www.lexonomy.eu/`) which contains Lexonomy's internal ID of the subentry:

```
<entry xmlns:lxnm="http://www.lexonomy.eu/">
  ...
  <phraseme lxnm:subentryID="54387">
    ...
  </phraseme>
  ...
</entry>
```

The `lxnm:subentryID` attribute (like all attributes and elements in the `lxnm` namespace) is hidden and never shown to the user in Lexonomy's XML editor, but it is always there. Based on these IDs Lexonomy replaces the element's content with the subentry's content (where by 'content' we mean all of its attributes, text nodes and child elements except the `lxnm:subentryID` attribute itself).

This formalism is functionally equivalent to *Simple Links* with *Replace On Load* behaviour as defined by the XLink standard. The example above could be rewritten in XLink as (hypothetical example, not actually used in Lexonomy):

```
<entry xlink:xlink="http://www.w3.org/1999/xlink">
  ...
  <phraseme xlink:href="54387" xlink:show="replace" xlink:actuate="onLoad">
    ...
  </phraseme>
  ...
</entry>
```

Additionally, Lexonomy's formalism for subentries is somewhat equivalent to a less well-known feature of Lexical Markup Framework (LMF) where multi-word entries can be independent entries which can then be linked to from specific senses of other entries via their ID.[3] Lexonomy's formalism is more general, however, because it allows the linking of any XML elements.

The process of inserting subentries into entries happens in Lexonomy at the time each entry is saved. For example, when a user saves an entry in which he or she has made a change to a subentry, the changed copy of the subentry is immediately propagated to all other entries where the subentry occurs, so that each entry always has the most recent copy of all its subentries. This means that Lexonomy does in fact keep several copies of the same subentry in several places, but the copies are synchronized with each other behind the scenes. To the user, it seems like there is only one central copy of the subentry in existence. The reason for this apparently unnecessary redundancy is to make sure that the entire contents of each entry is always easily[4] accessible for various searching and indexing operations, for example for XPath queries that 'look inside' subentries.

Because subentries can be recursive (there can be subentries inside subentries), the process of inserting subentries into entries is recursive too. In fact, there is no difference internally between entries and subentries, they are all stored as objects of the same kind and can contain other objects of that kind. Only the dictionary schema decides which of these objects are treated as entries and which as subentries: an entry's top-level element is also the schema's top-level element, typically called `<entry>`, while a subentry's top-level element is something else deeper down in the schema, such as `<phraseme>` or `<example>`. This even means that something can simultaneously be an entry and a subentry of other entries, if the dictionary is so configured.

6 Conclusion

The problem of multiword item placement has a solution which, in retrospect, seems obvious: to allow subentries to appear in more than one place in the dictionary. The only question is a technical one: how to make it possible.

The classical tree paradigm, as formalized in XML, is not expressive enough for this purpose because it does not allow elements to have more than one parent. On the other hand, the more generic graph paradigm, as formalized in Semantic Web RDF for example, is hugely over-expressive and is an overkill for this purpose because it departs too far from the familiar two-dimensional nature of trees. This paper has proposed a compromise, a tree-like structure which differs from XML in just one aspect: it allows elements to have more than one parent. This new paradigm – together with the new editing features in Lexonomy – allows lexicographers to continue working with trees most of the time, while occasionally 'breaking out' of the tree to share subentries among two or more entries.

[3] I am grateful to Adam Rambousek for this observation.

[4] By 'easily' we mean 'without additional processing'.

It is hoped that this will be one of the innovations that empower lexicography to depart from a situation where it is merely imitating on computer screens what it was previously doing on paper, and move to a smarter position where it can fully avail of the possibilities offered by the digital medium.

7 References

Aguado-de-Cea, G., Montiel-Ponsoda, E., Kernerman, I., Ordan, N. (2016) ‘From dictionaries to cross-lingual lexical resources’ in: *Kernerman Dictionary News,* 24, pp. 25-31.

Bogaards, P. (1990) ‘Où cherche-t-on dans le dictionnaire ?’ in: *International Journal of Lexicography,* 3(2), pp. 79-102.

Klimek, B., Brümmer, M. (2015) ‘Enhancing lexicography with semantic language databases’ in: *Kernerman Dictionary News,* 23, pp. 5-10.

LMF: ISO 24613:2008 Language resource management – Lexical markup framework, `http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=37327`

Měchura, M. B. (2017) ‘Introducing Lexonomy: an open-source dictionary writing and publishing system’ in: I. Kosem, C. Tiberius, M. Jakubíček, J. Kallas, S. Krek, V. Baisa (eds.) *Electronic lexicography in the 21st century: Proceedings of eLex 2017 conference,* Leiden, pp. 662–679, `http://www.lexonomy.eu/docs/elex2017.pdf`

Měchura, M. B. (2016) ‘Data Structures in Lexicography: from Trees to Graphs’ in: Horák, A., Rychlý, P., Rambousek, A. (eds.) *Recent Advances in Slavonic Natural Language Processing,* `http://www.lexiconista.com/raslan2016.pdf`

Polguère, A. (2004) ‘From Writing Dictionaries to Weaving Lexical Networks’ in: *International Journal of Lexicography,* 24(7), pp. 396-418.

Wiegand, H. E. (1989) ‘Der Begriff der Mikrostruktur: Geschichte, Probleme, Perspektiven’ in: Hausmann, F. J; Reichmann, O.; Wiegand, H. E; Zgusta, L. *Wörterbücher: Ein internationales Handbuch zur Lexikographie,* Berlin: de Gruyter, pp. 409-462.

XLink: XML Linking Language Version 1.1, The World Wide Web Consortium, `https://www.w3.org/TR/xlink11/`