Thoughts on using the Lexware system in 2018

Cam Webb (with comments by Tim Montler)

2018-12-18 (v. 4)

The Lexware system

When I first saw what a Lexware data file was at Jim's presentation a few months ago, I was surprised to see that it was just a plain text file, structured only by keywords ("band labels") and line order (plain text is simply a string of characters—ASCII and/or Unicode—with no formatting). As I have started working with the system, a number of pros and cons have occurred to me in regards to Lexware as a lexicography database.

The "Lexware system" I am referring to is this:

1. The Lexware (`.lex`) plain-text file.
2. A plain-text editor (see below).
3. A lex-to-product converter and other tools for processing the `.lex` files.

In the past, this converter would have been an installation of Bob Hsu's `LEXWARE` software. At the moment, the only option is a round-trip via Tim Montler: send him the files, and he sends back the product. Making this step no longer dependent on Tim is key, both for Jim, Joel, Evon, and also for Tim! This is where I may be able to help. I have Tim's `SPITBOL` code running on my laptop. But Jim and Tim and I have been discussing re-writing Tim's code to make it accessible to a wider range of users.

According to Tim, the tools that Jim has been using are a limited subset of the original `LEXWARE` toolset: i) A `.lex`-to-formatted-text converter (Tim does not use `LEXWARE` for this, but wrote his own conversion script in `SPITBOL`), ii) band-sorting (extraction of certain data elements), and iii) limited indexing (the `INVERT` functions of `LEXWARE`). The first two functions could quite easily be re-written; indexing may be a bit harder. The tool I would use is `awk`, a powerful, universally distributed tool designed for text-processing (https://en.wikipedia.org/wiki/AWK). I note that `awk` has been a choice for other lexicographers: http://www.billposer.org/Linguistics/Computation/LectureNotes/ParsingLexica.html.

The alternative "system" for dictionary work would be a choice of some pre-existing software. Options include:

- `Kirrkirr`(https://nlp.stanford.edu/kirrkirr/),
- SIL's tools:
  - Language Explorer (`FLEx`; https://software.sil.org/fieldworks/),
  - `Shoebox` (https://software.sil.org/shoebox/) and
  - `WeSay` (https://software.sil.org/wesay/),
- `TLex Suite` (https://tshwanedje.com/tshwanelex/),
- `WordSmith` (https://lexically.net/wordsmith/version6/index.html),
- `DictMaker` (<http://www.hostalive.net/dictmaker.html),
- `Lexique Pro` (<http://www.lexiquepro.com/).

You may or may not have experience with any of these. To varying extents, these tools contain database, editor and converter.

Pros of the Lexware system

- Infinitely flexible! You are not limited by the software designer's choices about elements you want to record. You can create a data element simply by minting a new band label. With this flexibility, Jim has customized the generic use of bands (a.k.a. data "fields", or "columns") for Dene languages. Evon now has a chance to build on this and add new elements for a Gwich'in dictionary (as we discussed). In addition, with some additional programming work one can invent new analyses and products.
- Also associated with plain-text data is ease of data entry. Once you know how to use an editing program, you can whiz around and edit/add data very fast. Most other dictionary programs would require many clicks to find the right "box" to type into.
- Perhaps the number one reason to use Lexware is *backward compatibility*: you can reuse and compare with Jim's files.
- The option for concurrent editing (see below). I guess that none of the existing desktop dictionary programs (above) permit this. You would have a single data file and have to send a master copy back and forth among collaborators, with the inconvenience and risks of data loss that this entails.

Cons

- Perhaps the number one issue is that there is no build-in validation with a plain-text file. If you misspell a band label or put something in the wrong order, you'll never know until the product comes out looking incomplete. This can be addressed by adding a stand-alone validator to the tool-chain (see below). Tim's `SPITBOL` converter incorporates a validator.
- Currently the dependence on Tim for producing an output, whether a dictionary or an analysis, is of course a major con. Hopefully, this barrier to use will be lowered soon.

Personally, I think storing data as Lexware-style plain text is a great choice. It fits with what we call the "Unix philosophy" where you handle data as plain text, with chains of simple, standard tools. But I'm not unaware of the drawbacks of this approach (and by extension of Lexware) for non-technical users.

In some comments he made on this document, Tim Montler confirms the value of using Lexware for Dene languages:

> "I think that none of the other lexicography software tools could be used with the Dene languages the way Jim envisions their organization. I use `Toolbox` (the successor of Shoebox) for my Salishan dictionaries, and it is very flexible; I've been very happy with using it for my Salishan dictionaries. But it has no sub-entry facility, and it would not be possible or easy to sort entries in `Toolbox` the way Jim does. `Flex` is very popular, but, as its creators acknowledge `Flex` is not flexible. Dene lexicography would have to be completely rethought to use `Flex`. `Lexique Pro` is based on the `Toolbox` format; I've used it for making a nicely formatted on-line dictionary (http://klallam.montler.net/d) from my `Toolbox` database. The others are simply not flexible enough for the organization that Jim uses. And I think that his organization is ideal for Dene languages. When I first started working with Jim, I tried to get him to switch to `Toolbox`. It wasn't long before we found out that it could not be done. The big problem is with the Dene verbs, which have a long string of various required prefixes."

The Lexware system represents a fascinating case of user-driven software design: The utility and importance of Lexware for linguists exists not in the software *per se*, but in the design of free-form text organization that requires no particular software skills. Building on Bob Hsu's original ideas, Jim Kari and Tim Montler have crafted a tool that is optimized to expose Dene language structure that could not have been done in a more constrained, generic, language software system. The role of the software to produce a formatted output is wholly secondary to the analytical insights of the linguist.

Tools

To run the Lexware system efficiently, a variety of tools are needed for end-users. Each one of these will need some learning; none should be beyond someone of average computer literacy, but sufficient training time needs to be included in estimates of the cost (time, and possible frustration) of using Lexware.

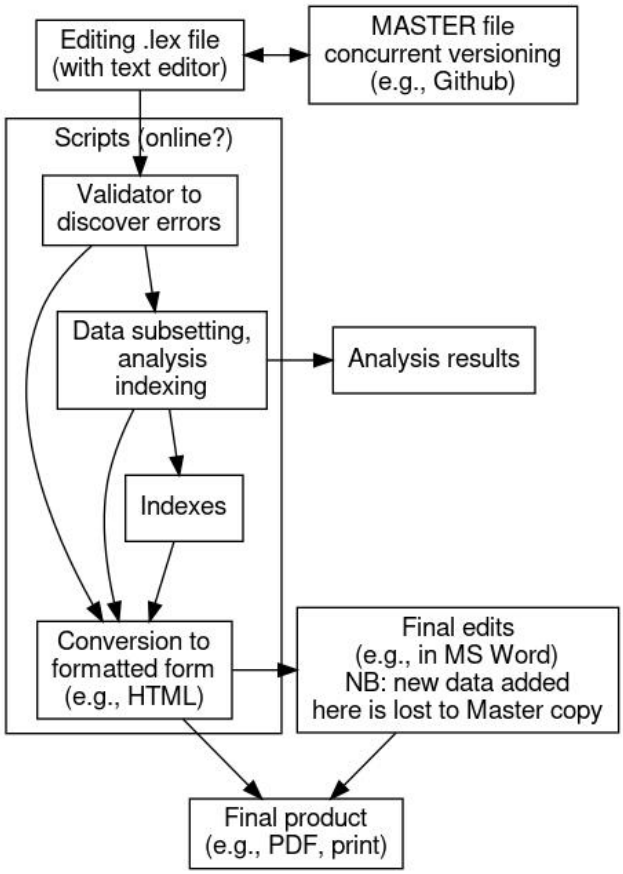

Figure 1: Lexware system production flow

Text editor

Since the core `.lex` data files are plain text, there are many options for creating and modifying them. As we discussed, one could use MS Word, or GoogleDoc, if one is careful: the text in the document can be exported back to plain text. However, while it may be attractive to use a tool one is familiar with, in general a Word Processor is not a good choice for plain-text editing. There is always a temptation to add formatting (font choice, bold, color...), which is not part of the plain-text data and which may even corrupt the data on conversion back to plain text. Additionally, word processors lack many of the powerful features of the many dedicated *text editors* created for software programmers (all software code is written in plain text). Such features include:

- Automatic coloration by syntax. E.g., the `.rt` band label and content can be made to automatically appear in a certain color.
- Powerful search (and replace). You may have heard Jim mention "regular expressions", or "regex": these are short phrases one writes containing special characters that match *variations* of a target phrase that cannot be found using simple character string matching.
- Line "folding", to show subsets of the lines without deleting the hidden lines.

Jim has a favorite editor (`EditPad Pro`, for Windows), I use another (`Emacs`, for Linux). Here is a list: https://en.wikipedia.org/wiki/List_of_text_editors.

Concurrent editing and versioning

Because a `.lex` file is in some ways itself simply a giant computer "program", the powerful tools that allow programmers to collaborate can be used to enable dictionary creators to collaborate: they can share a single master file and even edit the file simultaneously. And all changes can stored in *versions* so that present and past versions can be compared. Users of Google Documents will see the power of this co-editing paradigm, although as discussed above, Google Docs is not the most appropriate tool for editing `.lex` files. The most well-known online site that facilitates this concurrent editing is Github.com, which has become absolutely central to the work lives of many of the world's programmers.

Github use the `git` versioning system. The general work flow is: i) to *check out* a version of the file, ii) edit it on a local system, and iii) to *commit* it (or "check it in") back to Github. New changes can be easily visualized. If two people are using a file at once then their different changes are *merged* on committing.

Using these tools takes a bit of training and understanding, but the benefits are great. There are many graphical clients to help with using `git` and Github (see: https://git-scm.com/download/guis.

Text processing tools

At any stage of developing the `.lex` file, a user may want to manipulate the data, check the file, or make a formatted product. This is where Tim Montler has been needed by Jim, but where some new software development may free up Tim and allow for a range of new local solutions. The programming needed is relatively simple, and as I have been saying, finding one or several linguists who would be prepared to learn some basic programming skills would be a great boon for Dene dictionary work in Alaska.

Once developed, these tools could be run on a user's laptop, or online. Because of the slightly technical nature of running the scripts (via a *command line* or *batch file*), it may be easier for dictionary writers to have the scripts deployed as simple web applications. Users could visit a website and choose a *web form* that will perform the text manipulation needed. The `.lex` lines of a whole dictionary, or some smaller number of lines, could be pasted into a text box and with a simple click converted to formatted HTML, or some other product. The various tools that would need building are listed below.

Validation tool

As mentioned above, a major drawback to using plain-text files as a database is that errors can easily creep in. E.g., the misspelling of a band label. The way to counter this limitation is to regularly run a file through a *validator* program. If some rules about the structure of a `.lex` document can be established (e.g., the `eng` label must follow the `ex` label), then these can be encoded into the validator, which will then throw up warnings about the file being examined. The user can then correct the file and re-test it.

Data selection and sorting tools

Any number of subsetting tools for the master `.lex` file can be easily developed. E.g., the selection of loan words. The output file could be in `.lex` format, or might be a fully formatted file for printing.

While the line order (entry order) determines the default entry sort order, other entry sorts can be generates with scripts. Tim Montler describes these capabilities in the original `LEXWARE`:

> "Specialized sorting is a big part of Lexware. It's been used on scores of languages with exotic sort orders. That is the module called `HANDSORT`. A user adds a special 'handle' function that adds a generated bit of text as a sort key to the beginning of each entry. The simplest sort handles just substitute characters. For example, if you want 'ch' to sort as a single letter following 'c' in the alphabet (as in traditional Spanish sorting), substitute 'd' for 'ch' then 'e' for

'd' and so on. After sorting the handle is removed. You can also have one or more sub-handles to sort how something like 'a', 'á', etc. sort. In my Saanich dictionary I use two different sort orders—one for the main section and one for the root appendix—for two different orthographies. Jim has not used `HANDSORT` at all since I've been working with him, but Bob Hsu did a special handle and subhandle that sort, for example, 'b', 'm', 'p', and 'v' together, then subsorts them in that order. So all words beginning with those letters will be treated as the same letter of the alphabet, but if two words are identical except for those letters they sort in that order."

Product generation tool

The master `.lex` file is obviously not optimized for reading by end-users. Some subsetting (removing comments), and structural (sectioning, indentation) and typographic (bold, italics, symbols) formatting is needed to produce an easy-to-read product. This is what Tim Montler's main `SPITBOL` program does for Jim: it organizes the text and adds HTML tags. When the resulting file is opened in a browser it appears pleasingly formatted. This HTML file can also be opened in a Word Processor and final editing done. Note that new information added at this stage is not included in the master `.lex` file.

Indexing

The `INVERT`, `INVERT2` (Hsu 1990) and `INVERT3` (Hsu 2013) facilities of `LEXWARE` created "finderlists" — versatile indexes of language concepts and paradigms. To rewrite this functionality would be hard, and there is still the option to run the original PC version of `LEXWARE` on `.lex` files to generate Hsu's finderlists. However, it would be quite possible to generate simpler indexes using `awk` scripts. This may be part of the new development request.

References

- Hsu, R. 1990. Lexware Manual. Technical Report. URL: http://www.montler.net/lexware/LexwareManual-RobertHsu.pdf
- Hsu, R. 2013. Users guide to INVERT3. Technical Report. URL: http://www.montler.net/lexware/INV3MAN1.pdf