

## TshwaneLex Dictionary Compilation Software

from TshwaneDJe Human Language Technology

Reviewed by CLAIRE BOWERN, *Rice University*

TshwaneLex (<http://tshwanedje.com/tshwanelex/>) is a software program designed to facilitate the compilation of mono-, bi-, and multilingual dictionaries. Academic licenses cost €150, and commercial licenses, €1900. This review is based on a fully-working evaluation version of TshwaneLex 2.0. I spent a few days trying out the software, importing and exporting existing databases of mine, and starting a project from scratch.

In this evaluation, I wanted to answer three main questions:

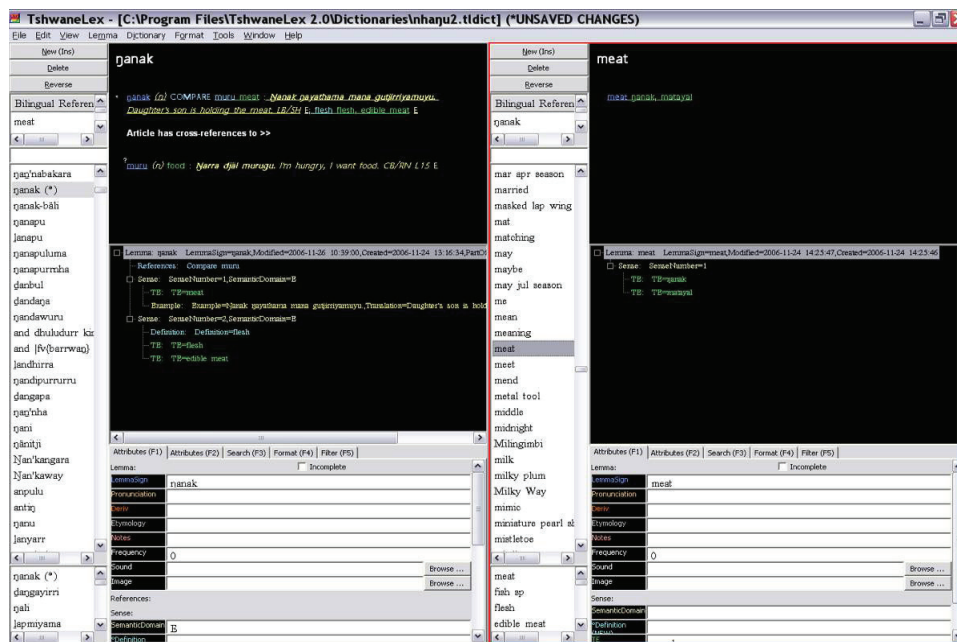
1. How easy would it be to start a dictionary project from scratch in this software, to maintain it, and to publish it? Would the software allow me to do everything I would want to do?
2. How easy would it be to convert an existing dictionary to this format, and what would be the advantage(s) of doing so? That is, why would one change from one's existing software to TshwaneLex?
3. How easy is it to exchange data between this program and other software that might be used in language documentation, such as transcription or interlinearization programs?

I conduct fieldwork on Australian languages, and much of my lexicographic work is done as part of larger projects in documentation and description. Therefore, I was particularly interested in the multilingual possibilities for TshwaneLex and in matters of data importing and exchange. My fieldwork data are also used for historical reconstruction, but in this test, I did not try to create a cognate database for historical data.

TshwaneLex uses XML markup, though one does not need to be familiar with XML to use the software. The setup of the database and navigation, however, will be more intuitive to people who have used databases with this type of structure. The interface includes windows that show a wordlist (a list of all lemmas in the database), a preview of the article (that is, the lemma and all associated information), a window showing recently visited lemmas, and a tree showing the information in the entry in a hierarchical structure. The software is fully Unicode compliant throughout. A screenshot of a sample entry is shown below.

For each database, there are attributes defined with respect to its different levels. The top level includes information about the language used in the dictionary (e.g., the target language and the language of the lemmas), as well as whether it is multilingual or monolingual. Other information is defined in relation to the individual entries in the dictionary (called “articles” in TshwaneLex), or with respect to subparts of the article, such as the senses. For example, translation equivalents (“glosses”) are part of the attributes of a sense; they are not an attribute of the lemma as a whole. After all, a particular example sentence might only be appropriate to one sense of a word, and not to another.

FIGURE 1: TshwaneLex sample entry



When TshwaneLex is opened, it is possible to choose from a local existing project or a shared database, or to start a new project. The latter brings up a dialogue box for general attributes for the dictionary as a whole, such as whether it will be multilingual or monolingual, and whether extra features should be enabled, such as options associated with Bantu noun classes. If one chooses the “new database” option, one is then presented with a blank database, into which it is possible to start entering headwords (lemmas) right away. There are also options to import wordlists and XML data (to be discussed below).

Although it is possible to start entering data fairly quickly, it is not possible to enter unstructured data. For example, it is not possible to enter a sense, an example, and a cross-reference without considering whether these pieces of information all belong to the same sense of the lemma. These pieces of information are not associated directly with the headword; rather, they are associated with an intermediate node in the entry tree structure. This avoids problems such as having an example associated with the wrong sense of a lemma.

The authors of a dictionary need to decide what types of information will be included. However, not all categories of the dictionary need be decided in advance in TshwaneLex. They can be added later (and their inclusion can be defined as optional or compulsory). This feature is important. It is very difficult to set up the final structure of a lexicographic database before beginning to work on the dictionary project, since many issues become clear only through entering the data and analyzing the language. This is a problem with Toolbox (<http://www.sil.org/computing/toolbox/>), where subsequent consistent modification of the structure of dictionary entries is very difficult. (Changing a template for a Toolbox

dictionary does not update existing records, so it is very easy to end up with inconsistent data presentation. Moreover, in Toolbox there is no database structure enforcement.)

Another convenient feature of TshwaneLex is the way it references other entries. Cross-references cannot be added until the referenced item exists in the database; this avoids orphaned links. (If a lemma is deleted, the cross-reference is also removed.) In the database, one is shown all articles that link in and out of the entry, making it easy to avoid circular references (that is, entries of the type *ḡanak*, see *ḡatha*, and *ḡatha*, see *ḡanak*). Multiple reference types exist in the basic setup, and it is easy to add further types where necessary. References are also automatically updated, which means that they reflect any change in the spelling of a lemma.

A validation tool checks for common errors, such as mismatched parentheses, required information that is missing (the elements of an entry that are required can be changed), circular references, duplicate translation equivalents, and a few other errors. It does not check example data, but that fault could probably be rectified fairly easily by using another program, such as *Linguistica*. (That is, one would create a wordlist out of the words in all examples and extract the words that do not appear as lemmas in the dictionary.) A future release will be able to perform this function.

One can use keyboard shortcuts to enter data for most common tasks (such as new article creation, attributes of the entry, etc.), although I found it difficult to use keyboard navigation alone, since the shortcuts are specific to certain menus. For example, there is a keyboard shortcut to add a new sense to the lemma, but it works only when the lemma level in the tree is highlighted. To accomplish this, you need to use the mouse. That combination of methods is probably efficient for enforcing a consistent structure, but it makes the software less convenient for those who do not use a mouse/trackpad.

TshwaneLex is very customizable: for example, it was easy to add image and sound fields to both examples and senses. I added a semantic domain to the senses and was able to filter entries based on it. Fonts, colors, and other aspects of the display were easily changed. It is possible to export to a variety of formats, including XML, CSV, and formatted text, and that is also customizable. That is, it is possible to produce formatted, camera-ready copy from a TshwaneLex database, just as it is from Toolbox, and the export style formatting can be easily adapted. The “masks” feature is also excellent, since it allows for different versions of the dictionary (e.g., those with large-print, with sound, without sound, etc.). An excellent feature of the software is that it recognizes that web, print, and computer dictionaries need potentially quite different layouts!

The multilingual dictionary allows the user to see both parts of the dictionary simultaneously. One can also either work on both sides simultaneously, or write the Language1 > Language2 part first and then convert it to a partial reversed dictionary for further work. (I have more to say on reversals below.) In general, I would have preferred more flexibility in window resizing and rearrangement, but what has been provided is adequate.

The program contains other useful features as well. For example, it is possible to add information about corpus frequency (although that needs to be compiled with an external concordance program). It would be possible to use TshwaneLex in combination with another program that would find collocations in corpus data.

TshwaneLex contains various housekeeping fields, such as box to check if the entry is incomplete. It also provides support for multi-user tracking, and has the ability to lock entries for editing. It also includes a statistics tool and a “ruler” application that estimates the distribution of initial characters and tells whether some parts of the dictionary are overrepresented—for example, whether one has fewer words beginning with *t*- than might be expected. Of course, it is up to the linguist to work out whether this is an accidental gap or a pointer to collecting more data.

In short, if I were starting a lexicography project from the beginning and making a self-contained dictionary, I would be very happy using TshwaneLex. I think there are a few situations where TshwaneLex might not be ideal, but that should not detract from its excellence as a piece of software that does what it is designed to do very well. It is stable (it never crashed in the testing I did, unlike a few other linguistic data programs I have tried recently), and it is better than its rivals for dictionary making. Also, at €150 it is a good investment when one considers the time it would take to make an adequate custom database using a program such as FileMaker.

The second matter I had in mind when reviewing TshwaneLex was whether existing projects could easily be converted to use with this software. For me, that meant converting Toolbox (=Shoobox) backslash-coded database files. I tried two Toolbox databases and worked intensively on one, a Yan-nhaju/English/Djambarrpuyŋu dictionary of about 2,800 items which was originally compiled with Toolbox v1.5.0. Significantly for the test, the database is a mess, since it was compiled in Toolbox by three different people. The records contain numerous fields which are not consistently ordered from record to record, and some of the entries are quite complex, with multiple senses, examples, references, and user-defined fields. It also contains duplicate entries and paradigmatic information. Therefore, it is an ideal database for testing new software. If TshwaneLex handles a database like this, it will handle anything!

As might be expected, importing the data was straightforward in principle, but less so in practice. I did end up getting most of the data into the test lexicon while (largely) preserving the structure of the entries. I converted the data in several batches as multiple tab delimited files, matching the headword each time, and combining entries piece by piece. It is possible to import XML files or CSV (comma-separated values, i.e., tabbed wordlists) into TshwaneLex. The XML export from Toolbox had errors and would not work, so I exported several sets of fields from Toolbox and converted them to CSV. I did three passes—first, with lemma headword, part of speech, and English gloss; second, with lemma and semantic domain; and third, with lemma and examples (Yan-nhaju, English, and source). Multiple glosses were assigned to different senses (as required) and to word in various semantic domains. Importing the examples and associating them with the correct subentries involved some problems, but it was possible.

These issues obviously raise the question of why someone might want to convert a working Toolbox database to TshwaneLex. There are a couple of reasons to do it, as well as some reasons not to. A prime candidate for switching is a project with multiple users. The multi-user functions and database merging capabilities for TshwaneLex are considerably better than (for example) Toolbox’s. On a multi-user project, an enforced data structure is especially valuable, and ease of merging entries would no doubt repay the time needed to convert the data to a new format. Another reason to switch software might be for the

web interface and flexibility of export to various formats. To be honest, I prefer the format that Lexique Pro produces (as an “off the shelf” export), but those needing custom export options would be better served with TshwaneLex.

In compiling a multilingual dictionary, it is necessary to compile either a finderlist, or to reverse the information in one part of the dictionary so that it appears in the other section. Thus far, no lexicography software has solved this problem. Toolbox allows the creation of a finderlist; however, the information included is quite limited. This is suboptimal for dictionaries for endangered languages, or for those designed for language learners, since having looked up a word in the finderlist, they must then check the word in the main part of the dictionary for grammatical and other semantic information. Given the already limited resources for endangered language dictionary production, there is a seldom time to produce a full reversal by hand. TshwaneLex has a reverse function; however, it does little more than produce a finderlist. Full reversal of existing lexical entries would have to be done by multiple exporting and reimporting, and then checking the entries. Part of the problem is that a reversal section can have several potential audiences. For example, if English speakers look up an English word in an English-French dictionary to find the French equivalent, they need to know the gender of a noun, the conjugation of a verb, and so on. However, if French speakers look up an English word to find a French equivalent, that information will not be necessary. The latter audience is served by a minimal reverse list, but the former audience is not.

There are some reasons not to switch to new software. First, if one’s current setup works well, there is no need to change. Another reason would be a lexicon that is heavily integrated into an interlinearization setup. TshwaneLex is not a general field management tool, but rather is built especially for lexicography. Corpus work, transcription, and interlinearization must be done in another program. The interlinearization feature is probably the main reason that fieldworkers keep using Toolbox. This program captures the way that people compile dictionaries and write grammars (and, more generally, document languages) simultaneously. Therefore, if the primary purpose of the lexicon is to feed into the interlinearization of texts, one is better off staying with Toolbox. (Note that there **are** other interlinear parsers, including ITE (<http://michel.jacobson.free.fr/ITE/>) and Alchemist (<http://linguistica.uchicago.edu/alchemist.php>), and a workflow could probably be set up that would allow the main database to be kept in TshwaneLex while still being able to interlinearize materials.)

This brings us to the third question. How easy it is to exchange data between TshwaneLex and other software? In an ideal world, data would never need to be entered anywhere more than once. The linguist could start with a transcript optionally linked to an audio file and interlinearize it, inserting and correcting words in the lexicon as work proceeds. The lexicon file would be suited both to being exported to a web-based (i.e., electronic) or print dictionary; the texts would be linked with the audio, but also interlinearized, and there would be a way to export texts while preserving the interlinearization for use in papers, PhD dissertations, and so on. There would also be a note-taking function that would allow filing of information snippets that one comes across in the texts. The linguist could export all the words in each text (i.e., use the basic concordance and wordlist functions too), and categorize and export all notes as well. The interface would allow easy data entry, import,

and export. Of course, it is not an ideal world, and there is no ideal piece of software to do this. Currently the only program that allows most of these features is Toolbox.

In summary, TshwaneLex does what it is designed to do, and does it very well. It is not a general documentation tool, and those who treat it as such will be disappointed. If one is starting a dictionary from scratch, TshwaneLex is an appropriate tool. It is possible to convert an existing database to this system, although there are both advantages and disadvantages in doing so.

<b>Pros:</b>	Structured data entry, minimal set-up required, multiple users supported, multiple export formats, fully Unicode compliant, highly customizable, easy to archive data in open source formats.
<b>Cons:</b>	Difficult to use with other documentation tools, keyboard-only navigation is difficult, no full reversal (although this is true of other programs too).
<b>Primary function:</b>	Dictionary compilation and editing; lexicography.
<b>Platforms:</b>	Windows XP.
<b>Open Source:</b>	No. While the software itself is proprietary, TshwaneLex works with open-source format data and files can be easily exported to open source formats.
<b>Proprietary:</b>	Yes, available from <a href="http://tshwanedje.com">http://tshwanedje.com</a> . Licenses from €150.
<b>Reviewed version:</b>	2.0.
<b>Application size:</b>	Unknown.
<b>Documentation:</b>	Manual that includes examples of applications; web page with detailed overview and sample web dictionaries.

Claire Bower  
bowern@rice.edu