Introducing TshwaneLex – A New Computer Program for the Compilation of Dictionaries

David JOFFE[°], Gilles-Maurice DE SCHRYVER[‡] & D.J. PRINSLOO[#]
DJ Software, Pretoria, SA[°], Department of African Languages and Cultures, Ghent University, Belgium[‡] & Department of African Languages, University of Pretoria, SA[‡ #]

1. Introduction

One would expect that it would be possible for a prospective dictionary compiler to walk into a local software mega-store and to merely buy a dictionary compilation program from the shelf, as is quite a reasonable expectation for finding a word processing package, OCR software or even voice recognition tools. This is however not the case. In most set-ups dictionary compilers and publishing houses all use either their own-compiled, locally networked, closed systems, or dictionary compilation is simply done on a word processor such as Microsoft Word or Corel WordPerfect.

No sophisticated SA-designed dictionary compilation software is available either on the local market or internationally. This is a major headache not only for individual dictionary compilers in South Africa but also for this country's *National Lexicography Units* (NLUs) tasked with the compilation of monolingual and bilingual dictionaries for the nine official African languages of South Africa. At the moment these units are left with no choice other than to compile their dictionaries with word processing software until such dedicated software becomes available, preferably a package adaptable to dictionary compilation for any language.

In this interim period some dictionary compilers are smart enough to *simulate a database* while working in a word processor by designing artificial records and fields with non-printing labels such as *lemma*, *definition*, *examples of use*, *part of speech*, etc. This is simply done by creating a template using hard page ends as the beginning and end points of a dictionary entry, simulating a record, and hard paragraph breaks for field delimiters. Thus all dictionary entries compiled this way have the same structure. Although this is a far cry from the real thing, it increases the potential success rate of eventual importation into a new dictionary program substantially.

In the past few years many individuals and institutions worked hard under the auspices of the *Pan South African Language Board* (PanSALB) to get the units established and up and running and to guide them on a continuous basis. However, a computer program for the compilation of dictionaries is still the crucial missing link in the entire lexicographic process. Initiative was taken in 1998 by D.J. Prinsloo to facilitate the compilation of such a program by bringing together computer experts, lexicographers and publishers in a so-called *SA Invitation Team* – an effort later abandoned in favour of the Swedish program, *Onoma*. When Onoma was discontinued

one year ago, a team consisting of the current authors quickly set out to create a brand-new and innovative software package for dictionary compilation.

It was envisaged that the development of a new sophisticated program for dictionary compilation should take at least five years, but the first version, *TshwaneLex 1.0*, is already imminent. TshwaneLex has its roots in Africa, appropriately linking *Tshwane* 'Pretoria' and its main function, *lexicography*, in a single acronym. TshwaneLex will also be marketed abroad. Dictionary compilers in the UK, Europe and Australia already show a keen interest in this program. Commercial aspects such as licensing of the software and conditions as well as methods of purchase still need to be finalised.

TshwaneLex is a software tool for all-purpose dictionary compilation, with some specific customisations for the African languages. The initial primary focus of development is to fulfil the requirements of the NLUs. The first version of TshwaneLex will not have built-in corpus tools or terminology management functionality. These capabilities will be considered for future versions of the software.

Some of the general 'guiding design principles' for TshwaneLex are:

- *User-friendly*: The compilation environment is intended to be as intuitive as possible, in order to minimise required training time and required computer literacy level.
- *Speed*: TshwaneLex is intended to provide the most 'streamlined' possible editing experience for lexicographers. All primary editing tools should be quickly accessible from the main interface, as well as by means of keyboard shortcuts.
- *Immediate output preview*: There should be an immediate preview showing how the article would appear in a printed dictionary.

Initial practical testing and development of the software is being done on a bilingual dictionary as well as a monolingual dictionary. SeDiPro, a bilingual *Northern Sotho – English* dictionary of the Department of African Languages at the University of Pretoria that was originally compiled on a word processor, was successfully 'transferred' to TshwaneLex and is currently being expanded with TshwaneLex. Monolingual Northern Sotho (Sesotho sa Leboa) data is used to test the use of TshwaneLex in the compilation of a major monolingual dictionary.

2. Overview

TshwaneLex can be used for the compilation of both monolingual and bilingual dictionaries. The user is presented with a 'language editing window' for each language in the dictionary. This language editing window provides the primary interface for editing the dictionary.

The language editing window consists of four primary parts, as shown in Figure 1.

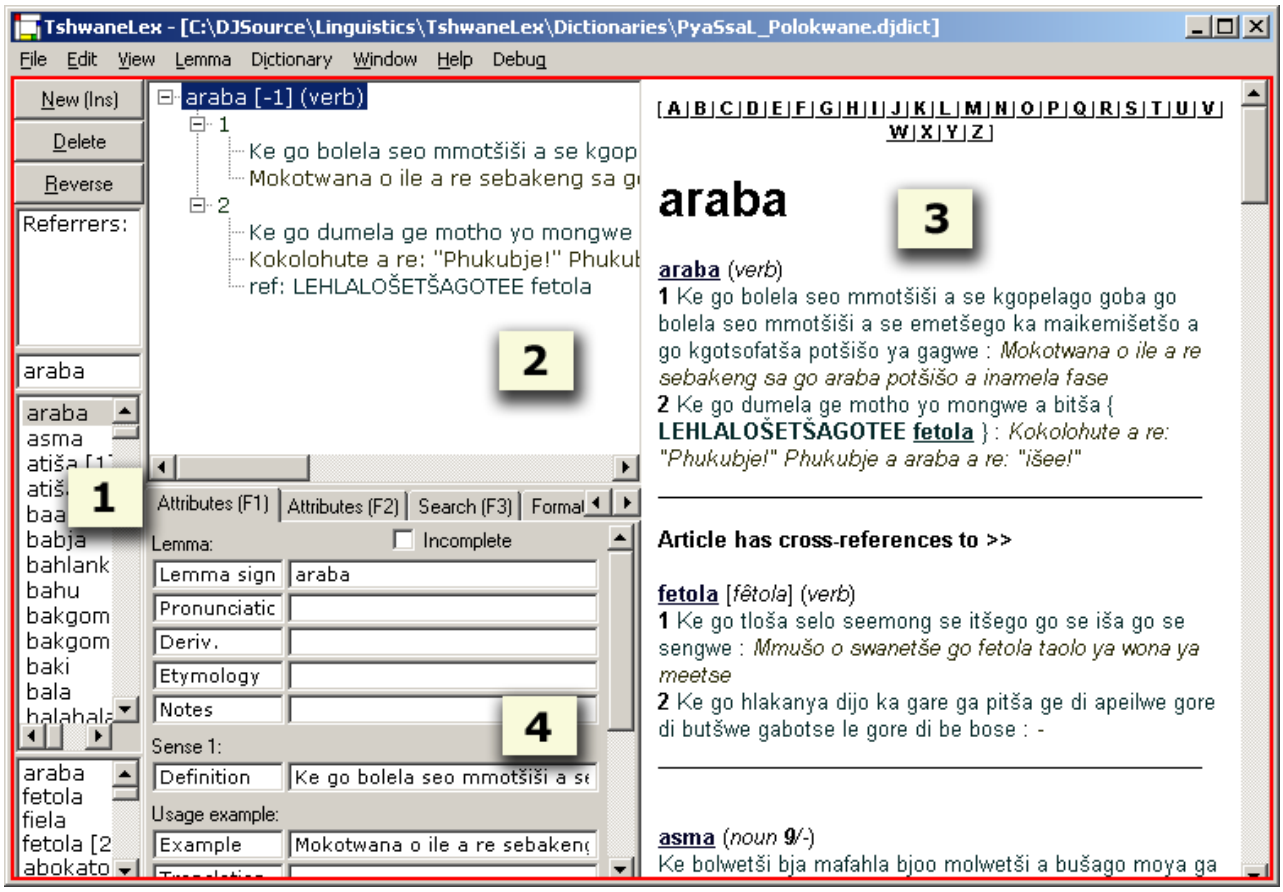

Figure 1: TshwaneLex screenshot showing the four main editing windows

1. **The lemma list**. A list of all lemmas for the language is shown at all times, from which the lexicographer can select which lemma to view or work on.
2. **The tree view control**. The 'tree view' shows the hierarchical structure of the currently selected article. This indicates the hierarchical relations of all senses, sub-senses, definitions, translation equivalents, usage examples, cross-references and combinations in the selected lemma. The tree view control also allows the structure to be edited, for example word senses may be created or deleted or moved around. Each element belonging to the selected article is represented by a 'node' in the tree. The specific text fields associated with that node are attributes of that node, and are edited in the tools window (cf. main editing window 4.).
3. **The preview area**. The primary purpose of the preview window is to show a preview of the article that closely resembles the output that would appear in a printed dictionary. Different elements of the article are displayed in different colours, for easy visual distinction. The article preview updates immediately whenever any changes are made to the article. One very useful feature of the preview area is that all cross-references related to (i.e. cross-references to and from) the currently selected lemma are also listed. The lemma signs of all cross-references

are displayed as clickable hyperlinks, allowing the referenced articles to be selected quickly.

4. **The tools window**. The tools window consists of a number of sub-windows. These are used for modifying attributes of an article, and provide other tools such as a text search function and a lemma 'filter'. These will be explained in more detail in §3.

When in bilingual editing mode, there are two language editing windows, one for each language. The two windows are arranged side by side, splitting the screen into two parts down the centre. This allows the lexicographer to work on both languages *simultaneously*. When working on a lemma in the source language, the lexicographer can immediately see how the translation equivalents of that lemma are treated in the target language, and vice versa.

3. The tools window

The tools window consists of five functionally distinct sub-windows, which are described in the following five sections.

3.1. Text attributes

This window contains text controls that allow the lexicographer to edit any of the text fields of the currently selected lemma. Depending on where one stands in the tree view, different (and appropriate) text controls are automatically generated.

Incomplete tag

This window also contains a checkbox labelled 'incomplete'. If a lexicographer is unsure about some aspect of the treatment of an article, and wants to return to it later, this checkbox can be used to tag the lemma as being incomplete. Lemmas that are marked as incomplete are automatically excluded from the output when the dictionary is being prepared for printed form.

3.2. Usage labels, parts of speech, noun classes

This window allows the lexicographer to specify the following attributes:

- usage labels (e.g. *formal*, *offensive*, etc.);
- part of speech tags (e.g. *noun*, *verb*, etc.);
- noun classes.

These may be applied to the lemma or to specific word senses within a lemma.

3.3. Search

This function allows the user to search the dictionary for a text string. Standard search options such as 'whole word only' and 'case-sensitive' may be selected.

3.4. Format and preview display options

An important feature of TshwaneLex is that the language of the metalanguage can be set. This simply means that the language for cross-reference types (e.g. *see*, *compare*, etc.), parts of speech and usage labels can be changed throughout the dictionary with just one mouse-click. This allows multiple dictionaries to be created from the same dictionary database for different markets. For example, two different versions of a bilingual Northern Sotho – English dictionary can be produced; one for English speakers, and the other for Northern Sotho speakers.

3.5. Filter

The filter allows the lexicographer to define criteria for selecting and viewing a subset of the articles in the dictionary. For example, the lexicographer can select to view "only those lemmas that are marked as incomplete", or to view "only those lemmas that do not have usage examples".

Fairly complex filters can be defined by combining the "include/exclude" filters and the "and/or" options. For example, in Figure 2, a filter is defined to show all articles that have cross-references but not definitions.

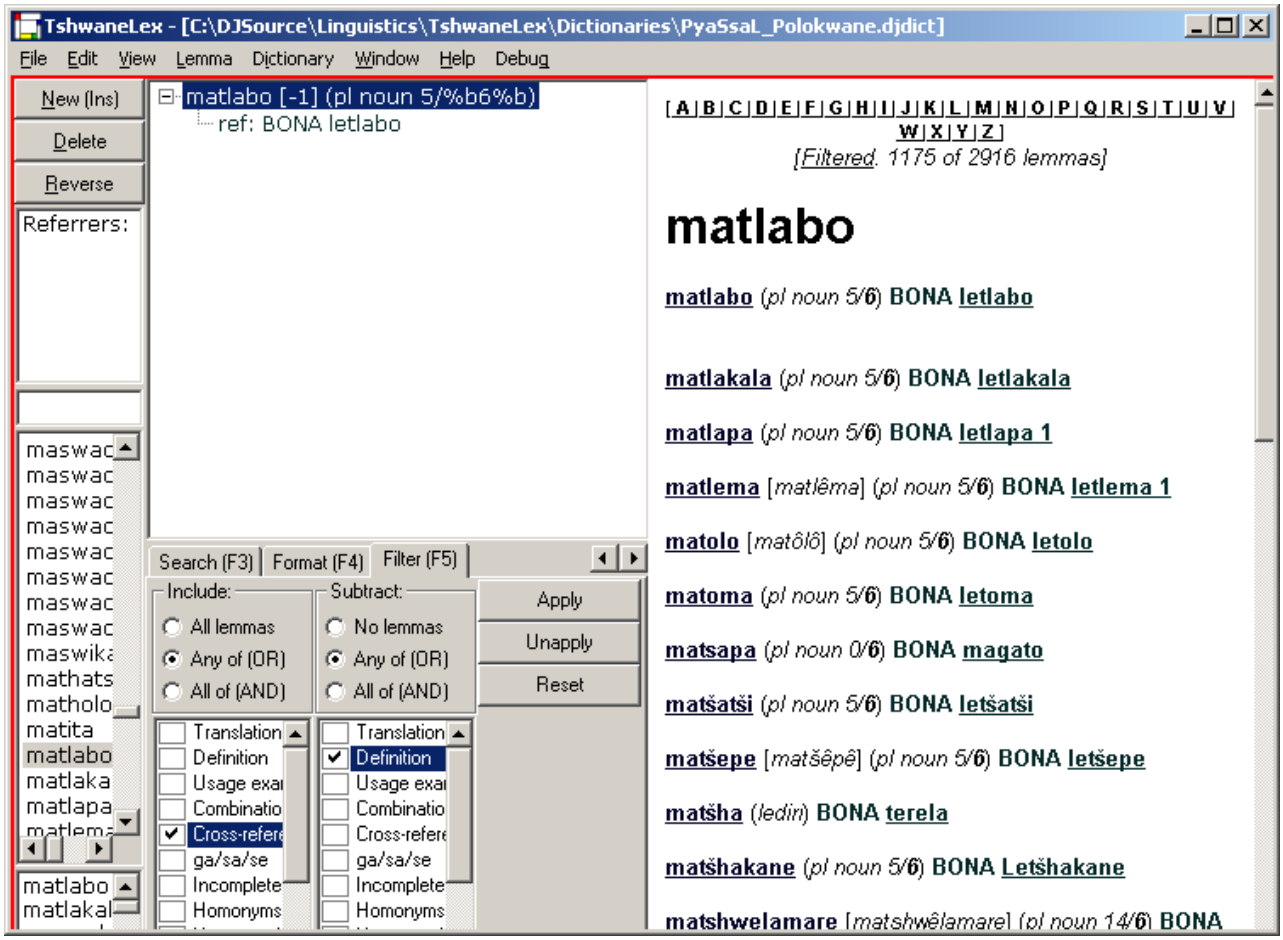

Figure 2: A screenshot demonstrating the 'filter' function (F5)

4. Bilingual editing features

4.1. Linked view mode

When in 'linked view mode', the language window for the target language automatically displays related lemmas when a lemma in the source language is selected, and vice versa. This is highly useful for quickly comparing the treatment of equivalent words on both sides of the dictionary.

4.2. Automatic reversal

TshwaneLex provides functionality to automatically reverse lemmas; that is, to automatically generate reversed lemmas in one language from the other. Either a single lemma can be reversed, or a full reversal can be done on all the lemmas in a language. The full language reversal has various options for how to handle multi-word translation equivalents. The lexicographer may choose to reverse single-word lemmas only, up to two-word lemmas, up to three-word lemmas, or the full dictionary.

Naturally, automatic lemma reversal is not all that is required to generate the treatment of words in the reverse side of the dictionary. Normally the lexicographer will still have to do work on these automatically created lemmas. For this reason, all lemmas generated by automatic reversal are marked as incomplete, indicating that they require further work from the lexicographer. Automatic reversal is thus not intended to *replace* manual reversal, but rather it is intended merely as a tool to speed up the lexicographer's work.

5. The cross-reference system

Automatic sense and homonym update

Internally, TshwaneLex stores the actual *structure* of cross-references, rather than storing a cross-reference as a 'dumb' text string. TshwaneLex uses internally unique identifiers for each referenced lemma or sense 'node'. This allows the system to automatically update the cross-reference should the target lemma sign, homonym number, or sense number of the cross-reference target change.

As an example, if the verb *araba* 'answer' contains a cross-reference to "**fetola :2**" (*fetola*, sense 2), and a lexicographer working on *fetola* decides to switch senses 1 and 2 around, then the cross-reference from *araba* will automatically update itself to "**fetola :1**" (*fetola*, sense 1).

6. Dictionary error checks

TshwaneLex can search for a variety of dictionary errors that lexicographers may make, such as duplicate definitions, duplicate translation equivalents, duplicate combinations, bracket nesting errors, redundant cross-references, redundant white-space and more.

7. Dictionary compare/merge

The dictionary compare/merge function allows a dictionary database to be compared to another dictionary database, with the possibility to merge new or modified lemmas into the main dictionary database. This functionality is particularly useful for situations in which lexicographers are split up geographically, and a high-speed network connection to the primary database server is logistically or economically infeasible. This is usually the case in all but the most developed countries. This feature allows the work done by remote teams to be periodically merged into the primary database.

This feature is also useful for comparing a dictionary against an older, previously backed up, version of itself; the lexicographer can thus easily see what changes have been made since the backup. This function is illustrated in Figure 3.

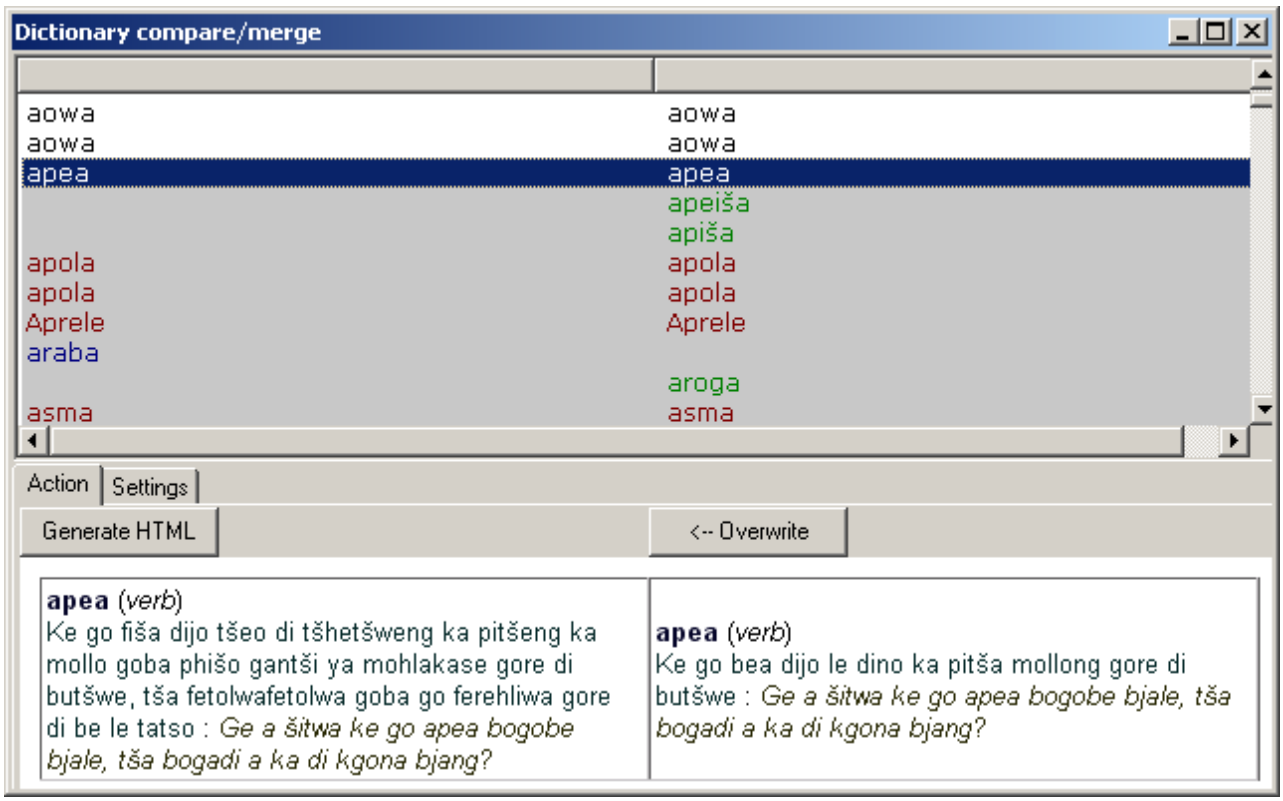

Figure 3: A screenshot of the dictionary compare/merge window

8. Backup system

Regular database backups are an absolutely essential part of any dictionary project. For this reason, TshwaneLex has some built-in functions to assist with the process of creating backups. This includes an option to automatically create backups at specified intervals.

9. Database interface and importers/exporters

TshwaneLex interfaces with a relational database server to access and modify the dictionary database. Multiple lexicographers can thus connect to the server and work on the dictionary simultaneously.

TshwaneLex uses the ODBC (Open Database Connectivity) cross-platform database interface standard to connect to the database server. This essentially allows TshwaneLex to use any ODBC-capable database software. ODBC drivers are available for all major database products on the market.

TshwaneLex can import wordlists and word frequency lists generated by corpora tools.

TshwaneLex has the following built-in exporters:

- XML (eXtensible Markup Language)
- RTF (Rich Text Format), which can be loaded into Microsoft Word
- Static HTML (HyperText Mark-up Language)

Some possible exporters and interfaces planned for future versions include the LaTeX typesetting system, a dynamic web page interface (for online dictionaries), as well as a file format and software interface for electronic dictionaries (e.g. CD-ROM based).

9.1. Custom importers/exporters

TshwaneLex has been built with a modular design that allows additional custom input/output interfaces to be created, should a particular user want to use something other than ODBC. This modularity allows customized importers to be developed should it be required to *migrate* an existing dictionary database from another system. Custom dictionary export (output) modules could also be developed.

References

Further information on acronyms/technologies mentioned in this paper is available at the following locations:

HTML. *HyperText Mark-up Language*. <http://www.w3.org/MarkUp/>

LaTeX. http://www.latex-project.org/

ODBC. *Open Database Connectivity*. <http://www.microsoft.com/data/odbc/default.htm>

RTF. *Rich Text Format*. <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnrtfspec/html/rtfspec.asp>

XML. *eXtensible Markup Language*. <http://www.xml.org/>