A repository of free lexical resources for African languages: the project and the method

Piotr Bański
Institute of English Studies
University of Warsaw
Warsaw, Poland
bansp@o2.pl

Beata Wójtowicz
Institute of Oriental Studies
University of Warsaw
Warsaw, Poland
b.wojtowicz@uw.edu.pl

Abstract

We report on a project which we believe to have the potential to become home to, among others, bilingual dictionaries for African languages. Kept in a well-structured XML format with several possible degrees of conformance, the dictionaries will be able to get usable even in their early versions, which will be then subject to supervised improvement as user feedback accumulates. The project is FreeDict, part of SourceForge, a well-known Internet repository of open source content.

We demonstrate a possible process of dictionary development on the example of one of FreeDict dictionaries, a Swahili-English dictionary that we maintain and have been developing through subsequent stages of increasing complexity and machine-processability. The aim of the paper is to show that even a small bilingual lexical resource can be submitted to this project and gradually developed into a machine-processable form that can then interact with other FreeDict resources. We also present the immediate benefits of locating bilingual African dictionaries in this project.

We have found FreeDict to be a very promising project with a lot of potential, and the present paper is meant to spread the news about it, in the hope to create an active community of linguists and lexicographers of various backgrounds, where common research subprojects can be fruitfully carried out.

1 Introduction

The FreeDict project was started by Horst Eyermann in 2000 and initially hosted bilingual dictionaries produced by concatenating (crossing) the contents of the dictionaries in the Ergane project (http://download.travlang.com/Ergane/), with Esperanto as the interlanguage. At first, the data was kept in the DICT format (Faith and Martin, 1997).

DICT (Dictionary Server Protocol) is by now a well-established TCP-based query/response protocol that allows a client to access definitions from a set of various dictionary databases. It provides data in textual form, but it also has the potential of providing MIME-encoded content. The clients can be free-standing desktop applications or they can be integrated into editors or web browsers. DICT web gateways also exist (see e.g. http://dict.org/).

The DICT format is a plain text format with an accompanying index file. The FreeDict-DICT interface initially used so-called "c5 files".[1] A c5 example of an entry from version 0.0.1 of Swahili-English dictionary is presented below.

```
abiria
     passenger(s)
```

Later on, the project adopted the TEI P4 standard (Sperberg-McQueen and Burnard, 2002) as its primary format, and with the help of its second administrator, Michael Bunk, created tools for conversion from the TEI into a variety of other dictionary platforms. A simple dictionary editor was also created. The change of the primary format was a very fortunate move, thanks to which we can today recommend the project as the possible home for free African language dictionaries big and small.

We are going to base our discussion on the Swahili-English dictionary, the first FreeDict dictionary encoded according to the guidelines of TEI P5 XML standard (TEI Consortium, 2007). The dictionary in its current form is an offshoot of a different project of ours that we decided to make available on a free license and in version 0.4 contains over 2600 headwords. We use it to demonstrate a possible process of dictionary de-

[1] C5 is the format used, among others, by the CIA World Factbook, where the heading is at the left edge and the contents are indented by 5 spaces.

Proceedings of the EACL 2009 Workshop on Language Technologies for African Languages – AfLaT 2009, pages 89–95,
Athens, Greece, 31 March 2009.

velopment, from the simplest to the advanced, machine-processable form.

2 From glossaries to rich lexical databases: the possible shapes of FreeDict dictionaries

A dictionary can begin its life at FreeDict as a simple glossary, with the simplest format possible, as shown in the made-up entry below:

```
<entry>
   <form><orth>alasiri</orth></form>
   <def>afternoon</def>
</entry>
```

The next example entry comes from Swahili-English xFried Freedict Dictionary, version 0.0.2, compiled by Beata Wójtowicz. That dictionary contained around 1500 entries of varied quality. It was based on a dictionary extracted from Morris P. Fried's *Swahili-Kiswahili to English Translation Program*, to which selected entries from the first FreeDict Swahili-English dictionary (compiled by Horst Eyermann) were added. That version also introduced information on parts of speech.

```
<entry>
   <form><orth>alasiri</orth></form>
   <def>   afternoon</def>
   <gramGrp> <pos>n</pos> </gramGrp>
</entry>
```

On the way to version 0.3, the entry looked as follows:

```
<entry xml:id="alasiri">
  <form><orth>alasiri</orth></form>
  <gramGrp><pos>n</pos></gramGrp>
  <sense>
    <def>afternoon (period between 3
     p.m. and 5 p.m.)</def>
  </sense>
</entry>
```

All the bracketed information was then turned into separate `<note/>` elements, in order to make the translation equivalents easily processable (see Prinsloo and de Schryver, 2002, for remarks on processability of translation equivalents). The change was performed by regex search-and-replace, roughly from `\((.+)\)</def>` into `</def><note type="hint">$1</note>`[2], with a subsequent review of all the new `<note/>` elements extracted by an XPath query.

Depending on the regularity of expressions in brackets, some additional words would be inserted into the search string, to be converted into `<note/>` elements of the appropriate type. Initially, only `@type="hint"` was used, as the most generic. At the moment, there are several more specialized type values, including `@type="editor"`, which contains editorial remarks that will not be shown to the user but will remain in the source. `@type`-less `<note/>` elements are used for quick localized communication between editors and are discarded by XSLT scripts when preparing the source version for release (they are also clearly marked by the CSS stylesheet that accompanies the dictionary, so that the editors can easily spot each note when reviewing the dictionary in a browser). FreeDict advocates the use of some other types of notes: recording the last editor of the entry, the date of the latest modification, and the degree of certainty, valuable in this kind of projects (where, e.g., some automated changes would set the certainty level to "low" and as such requiring editorial approval).

In the current version, 0.4, the `<sense/>` element looks as follows.

```
<sense>
   <def>afternoon</def>
   <note type="def">period between 3
      p.m. and 5 p.m.</note>
</sense>
```

This is what we decided to keep in version 0.4, exactly for the purpose of illustrating the possible development stages of dictionaries. In the next version, the `<sense/>` element will eventually attain the form currently (i.e., after September 2007) recommended by the TEI Guidelines for translation equivalents:

```
<sense>
   <cit type="trans">
       <quote>afternoon</quote>
       <def>period between 3 p.m. and
       5 p.m.</def>
   </cit>
</sense>
```

The `<quote/>` element holds the translation equivalent that can be an anchor for dictionary

[2] This is in fact a slight simplification of what has been done, made for the purpose of clarity. Naturally, the regexes have to be adopted to the circumstances (regularity of markup, regularity of expressions in brackets, the number of such expressions per single element content, etc.). Sometimes, an XSL transformation may turn out to do a better job, thanks to the many string-handling functions of XPath 2.0.

reversal or concatenation. The <def/> element above is not abused anymore and holds a real definition, i.e., an "explanatory equivalent", which may become a sense-discriminating note in the reversed dictionary.

We stress that each of the XML structures presented above (some of them admittedly bordering on tag abuse) conforms to the general P5 format and can be easily processed and published. In other words, dictionary editors are not forced to conform to the final format in order to see their work being used and commented on.

The next section presents another aspect of dictionary creation, where what matters is the ease of data manipulation and filling in the predictable information for the developer.

3 Plural forms: an illustration of automated creation of entries

At the stage of development at which brackets have been eliminated from translation equivalents, a relatively simple entry might look as shown below. This is an entry as created by a developer, to be further processed by XSLT.[3]

```
<entry>
   <form>
    <orth>adui</orth>
    <ref target="#maadui"/>
   </form>
   <gramGrp><pos>n</pos></gramGrp>
   <sense><def>enemy</def></sense>
   <sense>
    <def>opponent</def>
    <note type="hint">in games or
    sports</note>
   </sense>
</entry>
```

This entry is then processed and turned into the form presented below.

```
<entry xml:id="adui">
   <form>
    <orth>adui</orth>
   </form>
   <xr type="plural-form"><ref tar-
   get="#maadui">maadui</ref></xr>
   <gramGrp>
    <pos>n</pos>
   </gramGrp>
   <sense xml:id="adui.1" n="1">
    <def>enemy</def>
   </sense>
   <sense xml:id="adui.2" n="2">
    <def>opponent</def>
    <note type="hint">in games or
    sports</note>
   </sense>
</entry>
```

If the plural form does not exist in the dictionary, the script creates an entry for it:

```
<entry xml:id="maadui">
   <form>
    <orth>maadui</orth>
   </form>
   <gramGrp>
    <pos>n</pos>
   </gramGrp>
   <sense>
    <xr type="plural-sense">Plural of
    <ref target="#adui">adui</ref>
    </xr>
    <def>enemy</def>
    <def>opponent <note type="hint">in
    games or sports</note></def>
   </sense>
</entry>
```

The equivalents in this kind of automatically created entries for plurals will remain within <def/> elements even after we adopt the cit/quote system mentioned above in the discussion of *alasiri*. This is because <def/> elements are not anchors for dictionary reversal, and plural entries will be skipped by the reversal tools, unless the plural/collective form has its own unique meaning, as is the case with e.g. *majani*, which is morphologically the plural form of *jani* "leaf; blade of grass", but apart from that, it should also be glossed as "grass", and it is the latter form that should become a headword in the reversed, English-Swahili dictionary.

Another area where the XML format and tools give excellent results is text normalization. An earlier example shows unnecessary spaces in version 0.0.2: <def> afternoon</def>. Handling these required only the use of an XPath function `normalize-space()`, which strips all the unwanted whitespace characters.[4]

All the indexing is also done automatically. The indexing system in this particular dictionary is based on the shape of the headword, which it is easy to convert into form acceptable by the XML ID attributes (the XPath `translate()` and `replace()` functions are handy here). All

[3] The verbosity of XML markup can be overwhelming, but many XML editors feature content completion and can save the developer a lot of typing, the more so that TEI schemas are often part of the editor package.

[4] That version contained more traps for machine-processing, such as bracketed parts of words — sometimes this was done in a nontrivial manner, as in the entry for *adui*: <def> enemy(-ies)</def>. See Prinsloo and de Schryver (2002) for remarks on the non-friendliness of such space-saving devices.

the entries are first reordered alphabetically, then the script checks for homographs and assigns them appropriate indexes (e.g. `chapa-1` for the noun meaning "brand", `chapa-2` for the verb meaning "beat") and appropriate attributes used later in the creation of superscripts (suppressed in the plain-text DICT format). Multiple senses are also treated similarly — each receives its own `@xml:id` attribute and numbering (see the example of *adui* above).

We emphasize the fact that the encoding format makes it possible to reduce the developer's workload, with each stage of the dictionary enhancement being publishable. This allows one to work on the dictionary on and off, in their spare time.

4 Visualization of underlying structure

Some Swahili words are best lemmatized as stems. Version 0.4 of our dictionary does not yet display the differences between bound stems and free forms, but we will transfer this functionality (which we use in another project) to one of the future versions. This will make it possible for us to, e.g., add hyphens to bound forms and prepend "-a " to adjectives introduced by the "-*a* of relationship", adding extra structure visible to the end user, but ignored in sorting or queries.

Disjunctively written languages can be handled similarly. Kiango (2000:36) lists the following examples from Haya, discussing the problems surrounding alphabetization of nouns in print dictionaries:

```
a ka yaga      'air'
e m pambo      'seed'
o mu twe       'head'
```

The vocalic pre-prefixes should not be used for the purpose of arranging headwords, because if they are, all nouns end up under one of the three letters. Instead, class prefixes (*ka*, *m*, *mu*) should form the basis for alphabetization. In the XML/TEI format adopted by FreeDict, such problems are easily solved, either by using a separate element to hold the pre-prefix, or by the use of an appropriate attribute. The former solution is illustrated below.

```
<entry>
   <form>
    <orth extent="ppref">a</orth>
    <orth>ka yaga</orth>
   </form>
   <def>air</def>
   </entry>
```

The default value for the `@extent` attribute is "full", so it only needs to be mentioned where the value is different.

An AfLaT reviewer rightly points out that in an electronic dictionary, alphabetization is irrelevant. Indeed, the DICT format features separate "dictionary" and "index" files, and searching is done on the index file, which addresses the relevant portions of the dictionary file. The issue of alphabetization arises, however, in two cases: when preparing a print version of the dictionary, or when using the dictionary outside of the DICT system (this is a "working view" for the maintainers that can also be used as an out-of-the-box view for users).

In connection with the first case — preparing print/PDF versions of dictionaries — it is worth pointing out that conversion from TEI XML into various publication formats is made easy thanks to the open-source XSLT conversion suite maintained by Sebastian Rahtz (http://www.tei-c.org/Tools/Stylesheets/).

As for the "out-of-the-box preview", FreeDict dictionaries, by virtue of being marked up in XML, can be equipped with CSS stylesheets that make it possible to display the XML source in the browser, as if it were an HTML page. Here, because the user can search the page for the given form, alphabetization is not so relevant, but it can be handy, if only for aesthetic reasons. Below is a screenshot of a fragment of the CSS view of the file swa-eng.tei from version 0.4 distribution package, opened directly in the Firefox browser.

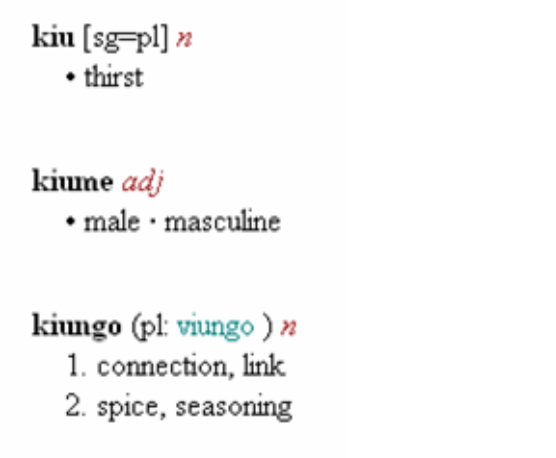

Figure 1: A CSS preview of the source XML

The CSS adds some text (e.g. "[sg=pl]", "(pl:", or sense numbering) and imposes visual structure onto the source XML. As can be seen in the entry for *kiungo*, we give precedence to formal

properties of headwords over the semantic distinctions, but other macrostructural decisions are obviously possible as well. The figure below shows one more CSS view, demonstrating subcategorization, treatment of notes (all the bracketed strings are contents of separate XML elements, with parentheses supplied by the CSS), as well as the treatment of homographs.

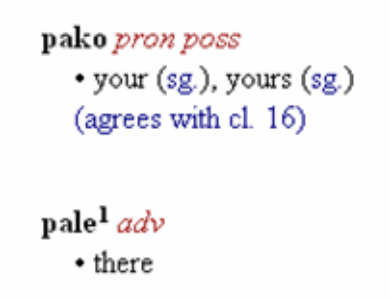

Figure 2: Subcategorization and grammar notes (another CSS view of the source XML)

Our use of colours (here: shades of grey) is also a function of the CSS, introduced mainly with the developer in mind, as a kind of an error-checking device.

5 Other possible enhancements of the microstructure

In section 3, we have illustrated a possible method of refining dictionary structure in an automatic fashion. Thanks to the many possible variations of the format, other features may be introduced stage by stage. They include, e.g., adding the corresponding plurals (illustrated above) or marking forms where the plural is the same as the singular, as done below by the use of the `@type` attribute. This example also illustrates the addition of cross-references to synonyms, where *eropleni* is linked to the second, inanimate, sense of *ndege* 'bird; airplane'.

```
<entry xml:id="eropleni">
   <form type="N">
    <orth>eropleni</orth>
   </form>
   <gramGrp>
    <pos>n</pos>
   </gramGrp>
   <sense>
    <def>airplane</def>
    <xr type="syn">(synonym: <ref tar
    get="#ndege.2">ndege</ref>)</xr>
   </sense>
</entry>
```

The `<form/>` element below illustrates the handling of alternative spellings of the noun *afisa*/*ofisa* "officer". Both headwords are used in retrieval.

```
<entry xml:id="afisa">
  <form>
    <orth n="1">afisa</orth>
    <orth type="variant"
          n="2">ofisa</orth>
    <ref target="#maafisa" n="1"/>
    <ref target="#maofisa" n="2"/>
  </form>
```

The above is the source as prepared by a developer. This is then processed, the plural forms are created if they do not exist in the dictionary, and the result is as in figure 3 below.

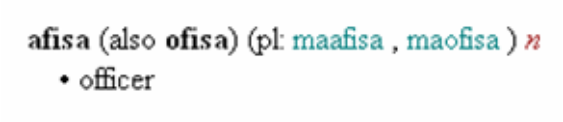

Figure 3: Illustration of alternative spellings/plurals (CSS view)

Other examples of what can be gradually introduced into the dictionary include addition of subcategorization information (illustrated by *pako* in Figure 2, where "pron" is the POS and "poss" is the content of the `<subc/>` element), addition of explicit noun-class- and agreement-marking, introduction of irregularly inflected forms, tables with inflection (linked to the appropriate stems), nested entries, and, obviously, the continuous improvement of lexicographic information (the arrangement and selection of senses, selection of headwords, an appropriate POS system).

Crucially, this system allows a developer to "publish early, publish often", and few of the enhancements mentioned here depend on others — developers are free to choose and to extend the dictionary at their own pace.

In our case, this is a gradual move towards structures of finer granularity, suitable for reversal (into English-Swahili) and concatenation with other dictionaries (we are going to use English as the bridge language for pairing the Swahili-English dictionary with English-* dictionaries).

6 Potential for the future

We wish to stress the potential that the encoding format and the entire project have for producing lexical resources for "non-commercial" languages, where funding and the time that the developers may spend on the dictionary are not always guaranteed. FreeDict dictionary development can proceed in stages, one can start with a

simple format and get the dictionary published on-line practically within days. The project has all the SourceForge publishing facilities at its disposal, together with bug/patch/etc. trackers and community forums. It also has a mailing list and a wiki that can serve to document some possibly difficult aspects of dictionary creation.

Thanks to the robust build system of FreeDict, creating a tarball containing a DICT-formatted dictionary and index is only a matter of issuing the "make" command with appropriate arguments, and submitting the resulting archive to the SourceForge file release system.[5]

FreeDict is the nexus for the following:

- XML, with its potential for creating well-structured documents,
- TEI P5, a *de-facto* standard taking advantage of this potential,
- the SourceForge repository as well as distribution and content-management network,
- the DICT distribution network: apart from being able to query DICT servers straight from the desktop, Firefox users can also take advantage of an add-on client that returns definitions for highlighted words on a web page,
- FreeDict tools (still under development for TEI P5) as means to manipulate dictionaries and to create, among others, the DICT format (usable directly from DICT servers and by other dictionary-providing projects, e.g., StarDict or Open Dict).[6]

Additionally, lexical resources submitted to FreeDict may undergo further transformations: reversal or concatenation, which means that work put into developing a single resource may well be re-used in developing others. Considering the possible re-use of lexical resources, they are expected to be prepared with a view towards clean exposure of translation equivalents (in the cit/quote system or at least by judicious use of separators and brackets).

The project has its own distribution system, in the form of GNU/Linux packages — for example, Kęstutis Biliūnas is the packager for Debian Linux and maintains a page tracking the usage of Debian-FreeDict packages;

Apart from the above, the content published by FreeDict is guaranteed to be free.

7 The costs of developing for FreeDict

An AfLaT reviewer suggested that we provide a measure of the effort required to develop a resource for FreeDict. We hope to show here that this is very much dependent on the quality and form of the resource and on how much time the dictionary creator is willing to invest into it. Crucially, given the open-source nature of the project, even a simple, small list of near-binary pairings of equivalents can be a) quickly made useful to e.g. the readers of web pages written in the given language, and b) extended by others into a more satisfying resource.

It may be that the effort needed to create language resources in e.g. Lexique Pro, an excellent free tool by SIL International (http://www.lexiquepro.com/) is smaller, but there are differences in that Lexique Pro is a Windows-only closed-source program whose native MDF (Multi-Dictionary-Formatter) format is not as flexible as XML and therefore cannot be processed by the many tools that handle XML and TEI in particular. Consequently, the perspectives for re-use of Lexique Pro dictionaries in computational linguistic applications are much smaller. To our knowledge, Lexique Pro does not make it possible for users to query words straight off web pages, which can be done thanks to dict, a Firefox add-on (http://dict.mozdev.org/). It admittedly has other advantages that make it a serious alternative.[7]

The ideal solution would be to have an editing front-end such as Lexique Pro coupled with the openness and modifiability of the data offered by FreeDict. Indeed, there are plans for creating a converter from the new LIFT interchange standard (http://code.google.com/p/lift-standard/) that the beta versions of Lexique Pro can read

[5] This is something that a dictionary creator need not bother about — submitting a TEI source of the dictionary to the mailing list is enough.

[6] The FreeDict build process provides targets for platforms other than DICT, e.g. the Evolutionary Dictionary (http://www.mrhoney.de/y/1/html/evolutio.htm) or zbedic (http://bedic.sourceforge.net/).

[7] We do not discuss professional commercial dictionary writing systems such as TshwaneLex (http://tshwanedje.com/tshwanelex/) because, despite the academic discounts, they may be out of range for the average developer. It is worth mentioning that the discounted versions of TshwaneLex come with the understandable "no-commercial-use" restriction, which is in conflict with either the GNU Public License or the nearly equivalent Creative Commons BY-SA license that all SourceForge resources must be under (cf. http://www.gnu.org/licenses/gpl-faq.html).

and write, to the version of the TEI format used by FreeDict. That would undoubtedly enhance the attractiveness of the project.

To sum up, developing for FreeDict minimally requires some basic knowledge of XML. Free XML editors exist (e.g. XML Copy Editor, http://sourceforge.net/projects/xml-copy-editor/) that can make editing easier by autocompleting the elements (inserting closing tags, suggesting elements and attributes that are allowed at the given place in the structure) and signalling encoding mistakes.

8 African Languages and FreeDict

A reviewer remarked that the link to African language technology in this paper appears only to be present in the examples. Indeed, FreeDict is not a project that focuses on African-languages — it is a project where African language resources can be hosted and quickly become useful to users. Given an opportunity, we will encourage researches dealing with other languages to join the project — which will hopefully result in the creation of more cross-language resources, especially given that the encoding format is not tied to any particular language and is able to easily accommodate features characteristic of practically any language.

During the session on "African Languages in Advance" at the 2008 Poznań Linguistic Meeting, where we presented our Swahili-Polish project and also mentioned FreeDict as the place where we wanted to donate parts of our test Swahili-English dictionary that would otherwise remain on our disks, we talked to an organizer of that session, Karien Brits, about the advantages that this project can have for some of her colleagues working on South-African languages. This is what encouraged us to move on with the FreeDict Swahili-English dictionary and we hope that others will also find this project, and the possibilities that it offers, attractive.

The FreeDict project has recently awoken after a period of lower activity, and at the moment, every week brings something new. Currently, as far as African languages are concerned, apart from Swahili↔English dictionaries, the project hosts very basic Afrikaans↔English dictionaries, and an Afrikaans-German dictionary (all of them in need of a maintainer).[8] We hope that FreeDict will become home to many African language resources, and, thanks to the possibility of dictionary concatenation, facilitate also the creation of many African↔European dictionary pairs as well as all-African bilingual dictionaries.

[8] Dictionary sources can be accessed from the "download" link on the project page: http://sourceforge.net/projects/freedict/. They can be queried online at e.g. http://dict.org/, by setting the appropriate language pair in the database.

Acknowledgements

We are grateful to three anonymous AfLaT-2009 reviewers for their helpful comments on the previous version of this paper, and to Michael Bunk for confirming that our version of the history of the project is close to reality. We also wish to thank Janusz S. Bień for turning our attention to the FreeDict project a few years ago.

References

DICT: http://www.dict.org/

Faith, Rik and Martin, Brett. 1997. A Dictionary Server Protocol. Request for Comments: 2229 (RFC #2229). Network Working Group. Available from ftp://ftp.isi.edu/in-notes/rfc2229.txt (last accessed on January 19, 2009).

FreeDict: http://www.freedict.org/, http://freedict.sourceforge.net/

Kiango, John G. 2000. *Bantu lexicography: a critical survey of the principles and process of constructing dictionary entries*. Tokyo: Institute for the Study of Languages and Cultures of Asia and Africa, Tokyo University of Foreign Studies.

Prinsloo, Danie J. and Gilles-Maurice de Schryver. 2002. Reversing an African-language lexicon: the *Northern Sotho Terminology and Orthography No. 4* as a case in point. *South African Journal of African Languages*, 22/2: 161–185

Sperberg-McQueen, Michael and Lou Burnard (eds). 2002. *The Text Encoding Initiative Guidelines (P4)*. Text Encoding Initiative, Oxford.

TEI Consortium, eds. 2007. *TEI P5: Guidelines for Electronic Text Encoding and Interchange*. Version 1.2.0. Last updated on October 31st 2008. TEI Consortium. Available from http://www.tei-c.org/Guidelines/P5/ (last accessed on January 19, 2009).