Alignment of resources on Egyptian texts based on XML

Mark-Jan Nederhof*
University of Groningen
Faculty of Arts
P.O. Box 716
NL-9700 AS Groningen
The Netherlands
markjan@let.rug.nl

Abstract

We present an XML format in which one can represent several kinds of resources on Ancient Egyptian texts, namely hieroglyphic encoding, transliteration, translation, and lexical annotation. The format is designed to allow automatic alignment of the resources. We report on a prototype of software for processing this format, which has been in successful operation for over one and a half years.

1 Introduction

Let us consider the following situation. One wishes to study a text, say The Eloquent Peasant. Several resources are available. There is the transcription into hieroglyphic of the four known hieratic manuscripts [5], we have several books containing translations [3, 6, 7], and scores of grammar books that contain isolated examples taken from this text.

For studying a text on which few publications exist, one may open, say, two books containing resources on that text, and use the index fingers of both hands to study the books in parallel, matching e.g. the hieroglyphic form of the text with a translation. However, for a text such as The Eloquent Peasant, just two index fingers do not suffice, and one is restricted to study only two or three resources at a time instead of studying all available resources in parallel.

This paper discusses AELalign, an XML format, designed to alleviate this problem.[1] In this format, we may represent several types of resources on Egyptian texts, in such a way that software may automatically align them on the screen, so that the user may study these resources at the same time without continuously having to switch.

An important property of our approach is that different resources can be compiled independently and remain physically separated in different files, which leads to maximal

*Supported by the Royal Netherlands Academy of Arts and Sciences. Secondary affiliation is the German Research Center for Artificial Intelligence (DFKI).

[1] For XML, see http://www.w3.org/XML/. There is another ongoing project in Egyptology involving XML that may overlap with the format discussed here, viz. METEOR, mentioned at http://www.oi.uchicago.edu/OI/PROJ/XSTAR/XSTAR.html.
At this point, a comparison is impossible to make however, since during its developmental stage, access to that project is restricted to a small consortium of participating universities (p.c. with Janet H. Johnson).

flexibility. For example, the hieroglyphic encoding from [6] may be in one file, and the translation from [3] may be in another, and running a program with the two respective file names as arguments is all that is needed to align these two resources and view hieroglyphic and translation beneath each other on the screen (or to print them in this form on paper).

For over one and a half years now, we have a prototype of software in operation that is able to perform the alignment of resources in the AELalign format. This prototype was programmed in Perl and has been tested on an application involving the AEL email list: subscribers to the list may submit their interpretations of parts of The Eloquent Peasant electronically, using a simplified form of AELalign, called AELalight. This form is automatically translated to AELalign and aligned with other interpretations and with the hieroglyphic. The result of alignment is in the form of HTML pages with which one may conveniently compare the different interpretations. Using the same format, quotes from The Eloquent Peasant from grammar books have been gathered, sorted and aligned with the hieroglyphic, to allow easy access.[2]

Section 2 of this paper explains the AELalign format, and Section 3 discusses AELalight. We discuss an application involving lexical annotation in Section 4. Further information on AELalign can be found on the Web, at:

`http://www.dfki.uni-sb.de/~nederhof/AEL/align.html`

and AELalight is discussed at:

`http://www.dfki.uni-sb.de/~nederhof/AEL/alight.html`

2 The XML format

Figure 1 shows excerpts from an example file in the AELalign format. The first line of a resource is just there to tell us that the document is XML; the exact form of this line deserves no further attention here, and neither does that of the second line.

The first piece of real information in a resource is a section mentioning the person who created the file, the date of initial creation, and possibly the dates of subsequent changes. The name of the resource, which is part of the header, is typically a short string that represents for example the surname of the author of the resource. Optionally, one may also include the URL of the site where the XML file is located. Below that, in the header text itself, one can add a description of the resource, possibly including a bibliography.

In the body of the file, we may find several so-called *blocks* of text between pairs of tags such as `<texthi>` and `</texthi>`, `<textal>` and `</textal>`, `<texttr>` and `</texttr>`, and (not indicated in Figure 1) `<textlx>` and `</textlx>`. Here, `hi` stands for 'hieroglyphic', `al` stands for 'alphabetic', i.e. transliteration using the (extended) Roman alphabet, `tr` stands for 'translation', and `lx` stands for 'lexical'.

Although for expositional reasons the example file in Figure 1 contains blocks of several distinct types, it is often better that different types be physically separated into different files, which enhances reuse. For example, a translation of The Eloquent Peasant from say [3] could be expressed as an AELalign file containing a single block, with the translation enclosed in `<texttr>` and `</texttr>`.

A notable exception is when the different types of text are created together or are extracted together from the same existing source; consider in particular examples taken

[2] See `http://www.dfki.uni-sb.de/~nederhof/AEL/peasant/guest0.html`.

```
<?xml version="1.0"?>
<!DOCTYPE resource SYSTEM "AELalign.0.1.dtd">
<resource>
<created> Created by me today. </created>
<header name="My name" url="http://www.my_site/my_file.html">
This file contains transcription, transliteration and
translation of The Eloquent Peasant.
</header>
<body>

<texthi>
<coord version="R" pos="1.1"/>
z:A1*Z1-p-Z7-wn:n-i-n:p*Z7-E15-x:a-Z7-Y1:n-A1-r:n-.:f-
M20-t:y-A1-p*W[e]:n-M20-t:N23-H-U2-A-.:t-U32-N33:Z2-O49
<coord pos="1.2"/>
i*s-.:t-wn:n-N42:t-B1:f-U7[se,e]-[shade]r[ss]:[ss]t[ss]-A2-B1-r:n-s
[...]
<coord version="B1" pos="33"/>
D35-x:n:d-D56*[fit]D54-.:k-Hr-Z1-H*b*s-.:Z7[scale=0.8]-[fit]S28-Z2:A1
[...]
</texthi>

<textal>
<coord version="R" pos="1.1"/>
s pw wn(.w) ^xw.n-^jnpw rn=f sxtj pw n ^sxt-HmAt
<coord pos="1.2"/>
jsT wn Hmt=f ^mrt rn=s
[...]
<coord version="B1" pos="33"/>
n xnd=k Hr Hbsw=j
[...]
</textal>

<texttr>
<coord version="R" pos="1.1"/>
There was a man called Khunanup. He was a peasant of the Wadi Natrun,
<coord pos="1.2"/>
and he had a wife called Meret.
[...]
<coord version="B1" pos="33"/>
you will not step on my clothes!'
[...]
</texttr>

</body> </resource>
```

Figure 1: Example of a file in the AELalign format.

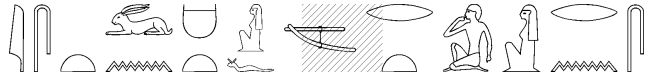
My name (R)	1.1 
My name (R)	1.1 | *s pw wn(.w) ˆxw.n-ˆjnpw rn=f*
My name (R)	1.1 | There was a man called Khunanup.
My name (R)	
My name (R)	*sxtj pw n ˆsxt-HmAt*
My name (R)	He was a peasant of the Wadi Natrun,
My name (R)	1.2 
My name (R)	1.2 | *jsT wn Hmt=f ˆmrt rn=s*
My name (R)	1.2 | and he had a wife called Meret.
⋮	
My name (B1)	33 
My name (B1)	33 | *n xnd=k Hr Hbsw=j*
My name (B1)	33 | you will not step on my clothes!'

Figure 2: The ideal output with the input from Figure 1.

from grammar books, where hieroglyphic, transliteration and translation occur closely together.

In Figure 1, note the tags of the form `<coord...\>`, which indicate positions in the text. Since The Eloquent Peasant is known in four manuscripts with distinct numbering of positions, we also need to indicate the versions, such as R and B1, that the positions refer to.

If the software is applied on a file such as that in Figure 1, it produces output as in Figure 2. The output shown here is somewhat idealized: there is not enough information present in Figure 1 that the software could induce that "*sxtj pw* [...]" and "He was [...]" should be aligned below the corresponding hieroglyphic. In such cases the software applies heuristics that usually give satisfactory, but not necessarily optimal, results. However, all other alignments shown in Figure 2 can be established automatically, based on the specified coordinates.

Below we will first discuss the four types of text that may occur in a block, and then we will explain the use of coordinates.

2.1 Hieroglyphic

Text between `<texthi>` and `</texthi>` is taken to be encoding of hieroglyphic. The software that processes AELalign renders hieroglyphic encoding into graphical form, as illustrated in Figure 2.

In the software as it is currently in operation, we assume encoding following the Manuel

de Codage [1], but we plan to replace this in the near future by the RES encoding scheme [4]. The examples shown in this paper also assume RES encoding.

Between pieces of hieroglyphic encoding, we find coordinates that help the software to align the hieroglyphic with other resources, as explained before. One may also use footnotes in the hieroglyphic encoding, e.g.:

```
M20 - t:y - empty[shade]<note>Perhaps restore <hi>M20 - t:y -
A1</hi>.</note> - p:n - n:N42 - t:B1 - f
```

This leads to a footnote marker within the graphical form of the hieroglyphic, and the footnote text itself is printed elsewhere. Note that nested within the footnote text, we may again find hieroglyphic encoding, this time enclosed in `<hi>` and `</hi>`, as well as translations and transliterations (see below).

Next to footnotes, one may also include normal text enclosed in `<no>` and `</no>` for short comments on the hieroglyphic that do not warrant a footnote, and that are to be printed in the Roman alphabet. A typical example is `<no>(sic!)</no>` behind the encoding of an incorrect spelling.

2.2 Transliteration

Text between `<textal>` and `</textal>` is taken to be transliteration, encoded in the usual way in ASCII, using `A` for aleph, `H` for "h-with-dot", etc. [1], and the character `^` indicates capitalization of the following letter. Software may render these letters in the usual Egyptological alphabet, although the current prototype does not have this possibility, since its output is HTML.

Just as hieroglyphic encoding, transliterations may include coordinates, footnotes, and normal text. Within footnotes, transliterations are enclosed in `<al>` and `</al>`.

2.3 Translations

Text between `<texttr>` and `</texttr>` is taken to be a translation. One may use `<al>` and `</al>` to delimit text representing transliterations inside of translations. A typical example is

```
handles of <al>sqb</al>-wood decorated with electrum
```

This specifies that `sqb` should be treated as composed of letters from the transliteration alphabet.

The translation may also include coordinates and footnotes.

2.4 Lexical information

Not shown in Figure 1 is the use of tags `<textlx>` and `</textlx>`, which delimit lexical annotation of a text. Such an annotation consists of a list of lexical entries, one for each word in the text, together with coordinates as usual. A full example of a lexical entry, showing all possible attributes, is given in Figure 3. The user is free to use the attributes as he wishes, and to use any combination of attributes and leave others unspecified. The intended use however is as follows.

The attributes `text..` refer to a phrase in the text at hand. One may indicate the hieroglyphic, the transliteration, the translation, some indication of the orthographic or syntactic form of the phrase, any combination of these, or none at all.

```
<lx
 texthi="R8-O6"
 textal="Hwt-nTr"
 texttr="Heiligtum"
 textform="honorific transposition"
 cite="Dictionary of Ancient Egypt"
 keyhi="O6"
 keyal="Hwt"
 keytr="enclosure"
 keyform="noun, fem, sing"
 dicthi="R8-O6-X1:O1"
 dictal="Hwt ntr"
 dicttr="temple"
 dictform="dir. genitive"/>
```

Figure 3: Example of a lexical entry.

The value of `cite` refers to the dictionary. This attribute is typically omitted if the only objective of the lexical entries is to develop ones own word list.

The key is intended to uniquely identify an entry in a dictionary such as [2]. As key one may include any combination of hieroglyphic, transliteration and translation. One may also specify aspects of the form of the key, such as gender and verb class. However, in most cases it suffices to have the key consist of the transliteration and the main translation. E.g. one dictionary entry may have a key consisting of *jrt* and "duty", another may have a key consisting of *jrt* and "eye". In rare cases one needs additional information on the grammatical function to distinguish between two entries in the dictionary.

By means of the values for `dict..`, one may provide a relevant phrase that is found in the dictionary under the specified key. Another typical example is:

```
<lx
 textal="wSd=f sw r mdt"
 texttr="he invites him to speak"
 keyal="wSd"
 keytr="address"
 dictal="wSd r mdt"
 dicttr="invite to speak"/>
```

One possible use of lexical annotation is discussed in Section 4.

2.5 Coordinates and alignment

Alignment is admittedly the most complicated aspect of AELalign, but also the most central. Alignment is the process of matching together the related parts from different resources on the same text. Alignment is essential for allowing us to study various resources simultaneously instead of one after the other, as outlined in Section 1.

There are two kinds of tag that are necessary for alignment. They generally appear in the following forms:

```
<coord version="version" pos="position"/>
<align version="version" pos="position"/>
```

For The Eloquent Peasant, *version* may for example be B1 or R, which refer to different manuscripts of the same text, and *position* indicates a position within a version of the text, and may for example be `27` or `7.6`. If the version is omitted from a coordinate tag, it is taken to be the same as that specified at the previous such tag.

One may also indicate different numbering schemes for one and the same version. Note e.g. that [3] assigns numbers to positions according to an older numbering scheme than that used by [6], both for B1 and R.

Before we can explain the meaning of coord(inate) and align(ment) tags, we first have to introduce the concept of *stream*. We define a stream to be the text of one single type (hieroglyphic, transliteration, translation, or lexical annotation) for one single version that occurs in one single resource. This means that one stream cannot contain information from different resources (i.e. different files), nor can it contain information for different versions, nor can it contain different types of text.

The most obvious use of coordinate tags is within blocks. Such a tag then specifies that the following text belongs to a certain version, and occurs just after the specified position. This information is valid until we encounter the next coordinate tag in the block, which specifies the next position within the same version, or a different version and some position therein.

One of the constraints on alignment of streams is now that all occurrences of a certain position in a certain version should be aligned beneath each other in the output. E.g. in Figure 1 there were three occurrences of position 33 of version B1. The output in Figure 2 aligns these three occurrences beneath each other, so that the hieroglyphic, transliteration and translation that follow the position in the three streams can be compared.

In most cases, a coordinate tag specifies a definite location in a stream. There are however also cases where we cannot distinguish a definite location in a transliteration or translation that corresponds to a position in the hieroglyphic. For transliteration a typical case is honorific transposition, when the beginning of a new line is between transposed signs. E.g. `sA ^ra`, "Son of Re", could be written as `^ra` at the end of a line, and `sA` at the beginning of the next line. If this line is numbered e.g. 5, this can be encoded as `<coord pos="5"> sA ^ra </coord>`, which specifies that the beginning of line 5 is somewhere in the indicated range, and all definite occurrences of the same position, i.e. those specified by a single coordinate tag, are to be aligned with some location between the beginning of `sA` and the end of `^ra`.

This issue is even more pervasive in translations, due to differences in word order between Egyptian and modern languages. E.g. we might find the following transliteration and translation for the same part of the text:

```
jw.jn rf <coord pos="321"/> sxtj pn r spr n=f 8nw sp

And <coord pos="321"> this peasant came </coord> to plead with
him for an eighth time.
```

Position 321 does not correspond to a definite location in the translation, so instead we indicate a range of words, delimited by the open and close tags of '`coord`'.

If we encode an existing translation in which no definite positions in the text were indicated, we may also need coordinate tags containing non-empty text. E.g. in [6] we find positions in the margins:

See for yourself:
B1 280 the apportioner is robbing,
the appeaser making suffer,

which is expressed in AELalign as:

```
See for yourself:
<coord version="B1" pos="280"> the apportioner is robbing, </coord>
the appeaser making suffer,
```

For a single version of a text, such as B1, coordinate tags specify how to align the streams for that version, irrespective of whether they came from the same resource (i.e. same file) or from different resources. To specify alignment for different versions on the same text we need alignment tags. Such tags are used in streams for one version to indicate alignment with a position in streams of another version.

A typical example is given in Figure 4. Versions B1 and R contain alignment tags referring to each other. Note that alignment tags are not visible themselves in the output, they merely affect alignment. E.g. *wnm jt=j* in version R is aligned below the text just after position 43 in version B1, due to the alignment tag preceding `wnm jt=j` in the input file.

Usually one does not have to specify correspondences between versions as exhaustively as exemplified here. Furthermore, often one would only use alignment tags inside the hieroglyphic encoding of the respective versions. The translations and transliterations for the different versions are then aligned indirectly since they are linked to the hieroglyphic by means of coordinate tags.

In AELalign one may also specify anonymous positions, i.e. positions that lead to alignment just as normal positions, like 42 for B1 or 9.6 for R, but that are not visible as strings in the output of the software. By using anonymous positions, we may obtain alignment on a finer scale than what would be possible by using only 'natural' positions, such as numbers of lines or columns, which may be quite far apart.

Anonymous positions can also be generated automatically. This option is activated by specifying a coordinate tag outside a block with position "`@anon`". Every sequence of blocks of different types (hieroglyphic, transliteration, translation), introduces a new anonymous position, which is inserted at the beginning of each block. As an example, consider:

```
<coord version="B1" pos="@anon"/>

<textal> mAA m <coord pos ="279"/> Hr=k </textal>
<texttr> See with <coord pos ="279"/> your own eyes: </texttr>

<textal> psSw m awnw </textal>
<texttr> the arbitrator is a cheat, </texttr>
```

This is equivalent to:

```
<coord version="B1"/>

<textal> <coord pos="@dummy1"/> mAA m <coord pos ="279"/> Hr=k
</textal>
<texttr> <coord pos="@dummy1"/> See with <coord pos ="279"/> your
```

```
<coord version="B1"/>
<textal>
<coord pos="42"/> mk wj r nHm aA=k sxtj
<align version="R" pos="9.6"/> Hr <coord pos="43"/> wnm=f Sma=j
</textal>
<texttr>
<coord pos="42"/> ‘Look, I shall take away your donkey,
peasant, <align version="R" pos="9.6"/> because
<coord pos="43"/> it ate my barley.
</texttr>

<coord version="R"/>
<textal>
<align version="B1" pos="42"/> mk Hm aA=k
<coord pos="9.6"/> Hr <align version="B1" pos="43"/> wnm jt=j
</textal>
<texttr>
<align version="B1" pos="42"/> ‘But look, your donkey
<coord pos="9.6"/> is <align version="B1" pos="43"/> eating my barley!
</texttr>
```

My name (B1)	42	*mk wj r nHm aA=k sxtj*		
My name (B1)	42	‘Look, I shall take away your donkey, peasant,		
My name (R)		*mk Hm aA=k*		
My name (R)		‘But look, your donkey		
My name (B1)		*Hr*	43	*wnm=f Sma=j*
My name (B1)		because	43	it ate my barley.
My name (R)	9.6	*Hr*		*wnm jt=j*
My name (R)	9.6	is		eating my barley!

Figure 4: Example in AELalign of alignment between versions, and the resulting output.

```
own eyes: </texttr>

<textal> <coord pos="@dummy2"/> psSw m awnw </textal>
<texttr> <coord pos="@dummy2"/> the arbitrator is a cheat, </texttr>
```

where `@dummy1` and `@dummy2` are anonymous positions.

3 AELalight

AELalight was originally created for an application involving the AEL (Ancient Egyptian Language) email list. The motivation was to allow participants to submit their interpretations on a text, consisting of transliterations and translations, which should then be automatically translated to AELalign and aligned with interpretations from other participants and with the hieroglyphic, in the form of HTML pages. In this way, the different interpretations could be easily compared.

One constraint was that the participants should be able to submit their interpretations in a form that was close to the non-computer-readable form they would normally use. This meant that XML tags and similar notation could not be used. The resulting format was called AELalight, since it can be seen as a 'light' variant of AELalign, at least with regard to its functionality, less so with regard to its syntax.

AELalight is now also used outside the context of the AEL email list however, in a somewhat generalized form allowing also hieroglyphic to be added. By means of AELalight, one may create files in the AELalign format with relatively little effort, since it frees the author from having to write the many XML tags that the AELalign format would require.

An example of part of a file in the AELalight format is provided by Figure 5. Note that coordinate tags are replaced by the much simpler positions between angled brackets, and positions may be absent from the translations, as in the case of 197 and 198 here. Corresponding transliterations and translations are separated by a line containing just a semicolon; we call such a combination of transliteration and translation a *paragraph*. By means of automatically generated anonymous positions, which were explained in the previous section, we obtain alignment of transliterations and translations belonging to the same paragraphs.

What is not shown here is that normal AELalign tags, e.g. alignment tags and notes, may also be used in AELalight. These remain unaffected in the translation from AELalight to AELalign. In this way, much of the power of AELalign is obtained, but with a much simpler syntax in general.

4 Building a word index

As already remarked above, XML formats such as AELalign are tedious to edit manually without specialized editors or automatic conversion from simpler formats such as AELalight. This holds in particular for the lexical entries described in Section 2.4. We have implemented a program that helps create AELalign files consisting of lexical entries, on the basis of an AELalign file containing transliteration for a text. The program has a graphical user-interface, implemented in Perl/Tk, and has access to an electronic dictionary, which allows lexical entries to be created by just a few mouse clicks. By means of the electronic dictionary, one can also verify the consistency of the lexical entries.

```
version = B1

<196> n wr js pw wr jm awn-jb
;
<196> A great one who is greedy is not (really) great.

tx pw <197> ns=k
;
Your tongue is the pointer (of the balance),

dbn pw jb=k rmnw=f pw sptj<198>=kj
;
your heart is the weight,
and your lips are its arms.
```

Figure 5: An example of the AELalight format.

A second program can be used to turn an XML file containing lexical entries for a text into an alphabetically sorted list, mapping dictionary entries and subentries to corresponding positions in the text. This index is output in HTML, which allows the use of hyperlinks from the word list to the HTML pages containing the text itself.

Acknowledgments

Chris Macksey and Mark Wilson contributed essential ideas, which were incorporated into the present XML format. Many thanks go to Geoffrey Watson for adapting his program mdc2html to simplify integration into AELalign, and to Jenny Carrington for providing hieroglyphic text that I used for testing the software. Deidre Lonergan helped me to get the XML format right.

References

[1] J. Buurman et al. *Inventaire des signes hieroglyphiques en vue de leur saisie informatique.* Institut de France, Paris, 1988.

[2] R. Hannig. *Grosses Handwörterbuch Ägyptisch-Deutsch: die Sprache der Pharaonen (2800-950 v.Chr.).* Verlag Philipp von Zabern, 1995.

[3] M. Lichtheim. *Ancient Egyptian Literature — Volume I: The Old and Middle Kingdoms.* University of California Press, 1975.

[4] M.-J. Nederhof. A revised encoding scheme for hieroglyphic. In *Proceedings of the XIV Computer-aided Egyptology Round Table*, July 2002. In press.

[5] R.B. Parkinson. *The Tale of the Eloquent Peasant.* Griffith Institute, Ashmolean Museum, Oxford, 1991.

[6] R.B. Parkinson. *The Tale of Sinuhe and Other Ancient Egyptian Poems 1940-1640 BC.* Oxford University Press, 1997.

[7] W.K. Simpson, editor. *The Literature of Ancient Egypt: An Anthology of Stories, Instructions, and Poetry.* Yale University Press, 1972.