ELAN-CorpA: Lexicon-aided annotation in ELAN

The CorpAfroAs project

One of the main objectives of CorpAfroAs[1] was to create a one-hour corpus of speech recordings in each language involved in the project, prosodically segmented, transcribed and morphosyntactically annotated. These corpora provide the basis for further cross-linguistic research.

From the outset, we were aware of a variety of tools that could help us perform at least some parts of the various tasks involved. Among these, ELAN[2] was the only tool capable of integrating the whole process. However, its capacity for multi-tiered annotations of media files lacks one significant function that is widely exploited by field linguists using Toolbox:[3] lexicon-aided annotation. So we decided to create this function in ELAN, since this software was open source.

The challenge was to manage an external lexicon from within ELAN. To do this, we decided to create two panels, one to interface with the lexicon and the other to display the segmentations proposed by the parser. Each of these areas had to be capable of responding to a mouse-click to validate the user's choices. A new tab, *Interlinearize*, was added to the main window, which

[1] http://corpafroas.tge-adonis.fr (9 September 2013). The project ended in June 2012 but the website and corpus are still in progress...

[2] ELAN is developed by the Max Planck Institute for psycholinguistics in Nijmegen; http://tla.mpi.nl/tools/tla-tools/elan/ (9 September 2013)

[3] Toolbox was developed by SIL International; http://www.sil.org/resources/software_fonts /toolbox (9 September 2013)

opens a new panel displaying a menu and two sub-panels *lexicon* and *segmentation* (see Figure 1).

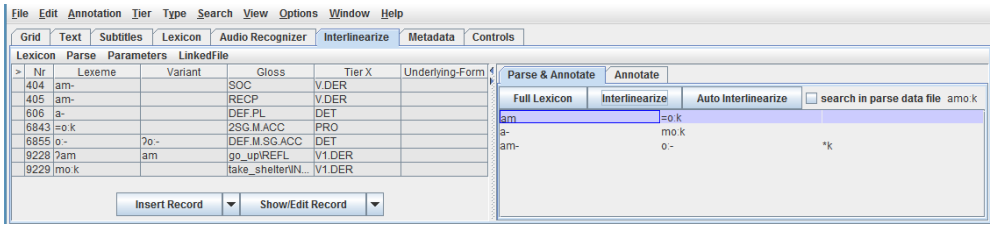

Figure 1: Segmentation of the word *amo:k*

The **lexicon panel** contains a grid to display the content of the lexicon, and two buttons to *Insert* a new record or *Edit* an existing one. Both these buttons let the user toggle between a record and a variant (i.e. the alternate form of a record).

The **segmentation panel** contains two tabs:

◆ ***Parse & Annotate***: this tab is where the full process of parsing the words into morphemes and annotating them takes place. The area displays the following options:

- ◊ *Full Lexicon* removes the filter masking the lexicon entries;
- ◊ *Interlinearize* launches the process of segmenting and annotating the word;
- ◊ A*uto-interlinearize* launches the same process without stopping at the next word if there are no ambiguities such as homonyms or multiple glosses to be resolved by the user;
- ◊ A checkbox allows the use of another lexicon containing already processed words to speed up the automatic process;
- ◊ A grid displays the segmentation proposed by the parser.

◆ ***Annotate***: this tab is for the process of annotation when the segmentation line already exists or if there is no need to parse the words into morphemes. The area displays the following options:

- ◊ *Annotate* simply annotates words or morphemes without segmentation;
- ◊ *Auto-Annotate* performs the annotation process without stopping at each word if there are no ambiguities such as homonyms or multiple glosses to be resolved by the user.

The interlinearize process

The *interlinearize* process begins by searching for the whole word in the lexicon. If it does not find it, it searches for the prefixes and suffixes recursively. In other words, it searches for the remaining part of the word after the isolation of a prefix or a suffix, until this remaining part is empty or no longer segmentable. The different possible segmentations of the word, relatively to the content of the lexicon, are then suggested to the user. If the remaining part of a segmentation is not empty, it is preceded by an asterisk to show that it is a guessed root not yet listed in the lexicon. The ***rootParser*** looks for the word or the remaining segment in the lexicon. The ***prefixParser*** and the ***suffixParser*** search all the possible lexicon entries (prefixes and suffixes) to find a match with the beginning or end of the target word or with the remaining segment.

These methods are applied on a string that represents the current segmentation of the word in the form:

(prefix-)*(remainingSegment)(-suffix)*

where the remaining part of the target word is surrounded by two special delimiters, e.g. internationalization => inter-[nation]-al-ize-tion

Specific lists from the lexicon

When the ELAN-CorpA lexicon is opened, various *arrays* are created depending on the type of lexicon entry: *rootForms*, *prefixForms*, *suffixForms* and *stemForms*. These lists are used by the parser to find a match with the analysed word or part of it.

The *stemForms* array contains the same type of lexicon entries as the *rootForms* array except that the entries appear in the lexicon preceded or followed by an underscore to show that they are not full words in the same way as roots are. Since the parser must ignore these underscores in the matching process, they are listed in a special array.

Underlying forms are lexicon entries that are analysed by the user as compounds of different entries from the lexicon. Since there can be underlying forms for each type of lexeme (roots, prefixes and suffixes), three corresponding arrays are created: *rootUnderlyingForms*, *prefixUnderlyingForms and suffixUnderlyingForms*.

The parser algorithm

The *rootParser* method is the entrance point to the *interlinearize* process. It applies to the whole word (in the script below: *annotation*), but throughout the process of morphological analysis, when affixes are gradually isolated

from the word, the remaining part is surrounded by two specific delimiters that let the parser know which part of the segmented word is to be processed (below: *X* or *Y*). The following algorithm explores all possible segmentations based on the lexicon affixes and roots.

```
rootParser (annotation)
    annotation = prefix* + X + suffix*
    /* X represents the remaining segment surrounded by 2 delimiters */
    if(X) not null
        foreach rootUnderlyingForm
            if match(X, rootUnderlyingForm)
                /* the root that matches is a compound of other roots */
                X => root*
                AddParseToList(prefix* + root* + suffix*)
                Exit /* only one underlying form for a root */
        foreach rootForm
            if match (X, rootForm) /* the rest is a word in the lexicon */
                AddParseToList(prefix* + rootForm + suffix*)
                foundRoot=True
        if foundRoot = False  /* the root is not in the lexicon
            /* search for suffixes */
            suffixParser(annotation, False, True)
                   /* search for prefixes */
            prefixParser(annotation, True, False)
    else /* annotation cannot be segmented anymore
        AddParseToList(annotation)

prefixParser(annotation, (boolean) foundSuffix, (boolean) foundPrefix)
    annotation = prefix* + X + suffix*
    foreach(prefixUnderlyingForm)
        if match (X, prefixUnderlyingForm)
            /* the prefix that matches is composed of other prefixes */
            X = (prefix+) + Y
            rootParser(prefix* + Y + suffix*) /* parse the remaining
                                                 segment Y */
    foreach prefixForm
        if match (X, prefixForm)  /* a prefix matches the beginning */
            foundPrefix=True       /* of the remaining segment X */
            X = prefixForm + Y
            partialSegmentation.add(prefix* + Y + suffix*)

    if foundPrefix = True
        foreach(partialSegmentation)
            rootParser(partialSegmentation) /* parse the remaining
```

```
                                                segments Y */
    else   /* no prefix available */
        if foundSuffix = True
            /* coming from a previous suffix segmentation
               or start process => search for other suffix in X */
            suffixParser(annotation, false, false)
        else /* matching no suffix nor prefix for the remaining segment X */
            guessedRoot="*X"
            AddParseToList(prefix* + guessedRoot + suffix*)

suffixParser(annotation, foundSuffix, foundPrefix)
    annotation = prefix* + X + suffix*
    foreach suffixUnderlyingForm
        if match(X, suffixUnderlyingForm)
            /* the suffix that matches X is composed of other suffixes */
            X => Y + (suffix+)
            rootParser(prefix* + Y + suffix*)
    foreach suffixForm
        if match(X, suffixForm) /* a suffix match the end */
            foundSuffix=True     /* or the remaining segment X */
            X => Y + suffixForm
            partialSegmentation.add(prefix* + Y + suffix*)
    if foundSuffix = True
        foreach(partialSegmentation)
            rootParser(partialSegmentation) /* parse the remaining
                                               segments Y */
    else  /* no suffix available
        if foundPrefix = True
            /* coming from a preceding prefix segmentation
            or start of process  => search for other prefix in X */
            prefixParser(annotation, false, false);
        else /* matching no suffix nor prefix for the remaining segment X */
            guessedRoot="*X"
            AddParseToList(prefix* + guessedRoot + suffix*)
```

AddParseToList

Adds a possible new parse if and only if it is not already in the list.

Displaying the results of the parser

The list of morphological segmentations of the target word is then displayed in the *segmentation panel* grid (see Figure 2).

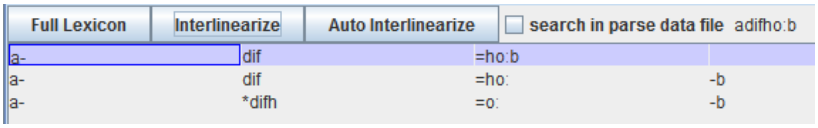

Figure 2: Possible segmentations of the word *adifho:b*

The user chooses the correct segmentation, then validates each segment of it with reference to the lexicon where one morpheme may have several values (either in terms of the gloss and/or the grammatical category). The lexicon displayed in the lexicon area is filtered so as to display only those entries that correspond to the morphemes of the various segmentations (see Figure 3):

Lexicon Parse Parameters LinkedFile

>	Nr	Lexeme	Variant	Gloss	Tier X	Underlying-Form
	606	a-		DEF.PL	DET	
	4302	=ho:		1SG.ACC	PRO	
	4303	=ho:b		when	CONJ	
	6847	=o:	=jo:	1SG.ACC	PRO	
	8945	-b	-w	M	INDF.DET	
	8946	-b	-w	REL.M	INDF.DET	
	9229	=o:		3SG.ACC	PRO	
	9230	dif		leave\PFV	V1	
	9231	dif		go	V1	
	9232	dif		lead	V1	

Figure 3: Lexicon for the segmented word *adifho:b*

Validating the morphemes of a segmentation

To select the correct annotations, a double-click on the morpheme in the segmentation area filters the lexicon to the corresponding entry with each value displayed on a different line. A double-click on the correct line in the lexicon validates the morpheme annotation. Then the next unit in the segmentation area is selected and the validation process continues. When the last morpheme is validated, these annotations are copied into the *annotation area*, underneath the target word.

Adding a new entry to the lexicon from the segmentation area

A remaining segment in the chosen segmentation line is preceded by an asterisk. The segment can be entered into the lexicon by right-clicking on it. A pop-up menu offers a choice between recording it as an entry or a variant. Depending on the choice, the *Insert Lexicon Data* window or the *Insert Variant* window opens.

If the parser fails to give the correct segmentation for a word, a right-click on an incorrect morpheme in the segmentation area lets the user access the *Insert Lexicon Data* or the *Insert Variant* window, and edit it accordingly. Then the *interlinearize* process may be relaunched to take this entry into account.

ELAN-CorpA Lexicon Structure

The lexicon used in the *interlinearize* process is an XML file that is linked to ELAN through its associated *.pfsx* parameter file:

```
<pref key="Interlinear.LexiconSource">
   <String> C:\myProject\myELAN\LexBej.eafl</String>
</pref>
```

The ELAN-CorpA lexicon contains both lexical and grammatical morphemes. A lexical entry has three main elements: the citation form of the morpheme <*Lexeme*>, its contextual forms (variants and underlying forms) <*form*>, and its meaning(s) <*sense*>. These are organized in the lexicon structure according to this DTD (where initial uppercase elements have no child element):

```
<!ELEMENT lexicon (lexicalEntry+)                    >
<!ELEMENT lexicalEntry (Lexeme, form?, sense?)       >
<!ATTLIST lexicalEntry
   id     ID          #REQUIRED
   dt     CDATA       #REQUIRED                      >
<!ELEMENT Lexeme   (#PCDATA)                         >
<!ATTLIST Lexeme
   typ  (lem|stem|wf|pref|suff) "lem"                >
<!ELEMENT form (altForm*)                            >
<!ELEMENT altForm  ((WForm, morph?)*)                >
<!ELEMENT WForm (#PCDATA)                            >
<!ELEMENT morph (Segment+)                           >
<!ELEMENT Segment (#PCDATA)                          >
<!ATTLIST Segment
  ref IDREF          #IMPLIED
  ge CDATA           #IMPLIED
  rx CDATA           #IMPLIED                        >
<!ELEMENT sense (stuff?, Gloss*)                     >
<!ELEMENT Gloss (#PCDATA)                            >
<!ATTLIST Gloss
  lang   NMTOKEN  #REQUIRED
  tierX  CDATA    #IMPLIED
  der    CDATA    #IMPLIED                          >
<!ELEMENT stuff (#PCDATA)                            >
```

The lexicon is composed of lexical entries <lexicalEntry> which have an identifier number and a date of creation/modification. Each **<lexicalEntry>** has three elements:

- **<Lexeme>**: whose value is the entry form and which has a *type* attribute: *root*, *stem*, *wordForm*, *prefix* or *suffix* (typ = lem|stem|wf|pref|suf);

- ◊ **<form>**: contains the various alternate forms (*Variant*s) the entry can present in text, and possibly their *Underlying form*s;
- ◊ **<sense>**: contains the various *Gloss*es of the entry, and their associated categories (*tierX).*

The **<form>** element may contain various alternate forms <altForm>.

Each **<altForm>** element:

- ◊ must have a word form **<WForm>** which contains the value of that alternate form.

  This element may have a ***der*** attribute which would identify the derivation of this alternate form with regard to the entry form;
- ◊ may have an underlying form **<morph>** composed of various lexicon entries listed in ****.

```
<form>
    <altForm>
      <WForm>yór</WForm>
      <morph>
        <Segment ref="14">H-</Segment>
        <Segment ref="6637">yor</Segment>
      </morph>
    </altForm>
  </form>
```

The **<sense>** element contains the various glosses **<Gloss>** of the entry with their associated categories ***tierX***.

```
<sense>
    <Gloss lang="en" tierX="V2">like</Gloss>
    <Gloss lang="en" tierX="V2">wish</Gloss>
    <Gloss lang="en" tierX="V2">want</Gloss>
</sense>
```

Importing a Toolbox dictionary

It is possible to import a lexicon from Toolbox and save it as an ELAN-CorpA lexicon (see *Interlinearize menu*). The interface allows the user to associate ELAN-CorpA lexicon entities with their corresponding Toolbox fields. The most common associations are:

Toolbox		**ELAN-CorpA**
lexeme	\lx	Lexeme
gloss	\ge	Gloss
part-of-speech	\ps	TierX
alternate form	\a	Variant
underlying form	\u	Underlying-form

TierX versus Part of Speech

In the Toolbox lexicon structure, a part-of-speech \ps covers all the glosses \ge under it.

\ps V2
  \ge like
  \ge wish
  \ge want

So in cases of multiple glosses for a particular lexeme within the Toolbox parser, the category of a specific gloss is implicitly the part-of-speech that precedes that gloss in the lexicon. In the ELAN-CorpA lexicon, however, each *Gloss* has its own explicit category, *tierX*, which is not simply a part-of-speech category but can be a label of any kind necessary for the search engine to retrieve a morpheme. When a lexicon is imported from Toolbox, each *Gloss* will have its *tierX* attribute filled with the part-of-speech it depends on.

Problem of identifiers when importing the underlying form field (u)

In the ELAN-CorpA lexicon, the underlying form of an entry or its variant, **<morph>**, consists of a sequence of other lexicon entries ****, which must have identifier numbers (*ref* attribute). But, when importing, these *underlying* segments do not yet have identifiers, so they are imported with an empty *id* attribute. For this reason, during the *interlinearize* process on any word with an underlying form, the user will have to select from the lexicon the entries that match these different segments. This updates the *ref* identifiers of the underlying segments of that entry.

Residual fields

During importation, any Toolbox fields not associated to an ELAN lexicon field are saved inside an element labelled **<stuff>**.

Example of a simple root entry (no alternate form)

```
<lexicalEntry id="5738" dt="14/Jan/2010">
  <Lexeme typ="lem">taf</Lexeme>
  <form/>
  <sense>
    <Gloss lang="en" tierX="V2">take</Gloss>
  </sense>
</lexicalEntry>
```

Example of a prefix entry imported from Toolbox

```
<lexicalEntry id="6176" dt="19/Aug/2009">
  <Lexeme typ="pref">tuː-</Lexeme>
  <form />
  <sense>
```

```
      <stuff>
        \gn art.f.sg.n
      </stuff>
      <Gloss lang="en" tierX="DEF">DEF.SG.F.NOM</Gloss>
    </sense>
  </lexicalEntry>
```

Example of a variant** (altForm) **with multiple glosses

```
  <lexicalEntry id="702" dt="14/Jan/2010">
    <Lexeme typ="lem">ʔareː</Lexeme>
    <form>
      <altForm>
        <WForm>areː</WForm>
      </altForm>
    </form>
    <sense>
      <Gloss lang="en" tierX="V2">like</Gloss>
      <Gloss lang="en" tierX="V2">wish</Gloss>
      <Gloss lang="en" tierX="V2">want</Gloss>
    </sense>
  </lexicalEntry>
```

Example of a main entry with a derived form (altForm) ***and its function*** (der)
(tikʷ = go_down > atkʷan = go_down\PFV.1SG)

```
  <lexicalEntry id="6002" dt="14/Jan/2010">
    <Lexeme typ="lem">tikʷ</Lexeme>
    <form>
      <altForm>
        <WForm der="PFV.1SG">atkʷan</WForm>
      </altForm>
    </form>
```

```
    <sense>
      <Gloss lang="en" tierX="V2">go_down</Gloss>
    </sense>
  </lexicalEntry>
```

Note that, in the case of a verb (*V* in the *tierX* field), if there is a nominalised derivation, a new entry has to be made for it, rather than a variant. This is because, even if a variant of an entry may have a derivation category (*VN* in the *der* field) which is added to the *mb* and *ge* values of the entry in the corresponding tiers, the category that appears in *tierX* is that of the entry itself (*V*). So any variant must stay in the same category as that of the entry itself, otherwise a new entry must be created for it.

Example of an entry with a variant and its underlying form

```
<lexicalEntry id="6637" dt="12/May/2013">
    <Lexeme typ="lem">yor</Lexeme>
    <form>
      <altForm>
        <WForm>yór</WForm>
        <morph>
          <Segment ref="14">H-</Segment>
          <Segment ref="6637">yor</Segment>
        </morph>
      </altForm>
    </form>
    <sense>
      <Gloss lang="en" tierX="v">s'arrêter</Gloss>
    </sense>
  </lexicalEntry>
```

yór	
H-	yor
Inac-mv-	s'arrêter v

Exportation of the ELAN-CorpA lexicon to Toolbox

```
<lexicalEntry id="702" dt="14/Jan/2010">
  <Lexeme typ="lem">ʔareː</Lexeme>
  <form>
   <altForm>
    <WForm>areː</WForm>
   </altForm>
  </form>
  <sense>
   <Gloss lang="en" tierX="V2">like</Gloss>
   <Gloss lang="en" tierX="V2">wish</Gloss>
   <Gloss lang="en" tierX="V2">want</Gloss>
  </sense>
</lexicalEntry>
<lexicalEntry id="6637" dt="12/May/2013">
  <Lexeme typ="lem">yor</Lexeme>
  <form>
   <altForm>
    <WForm>yór</WForm>
    <morph>
     <Segment ref="14">H-</Segment>
     <Segment ref="6637">yor</Segment>
    </morph>
   </altForm>
  </form>
  <sense>
   <Gloss lang="en" tierX="v">s'arrêter</Gloss>
  </sense>
 </lexicalEntry>
```

```
\lx ʔareː
\a areː
\ge like
\rx V2
\ge wish
\rx V2
\ge want
\rx V2
\dt 14/Jan/2010

\lx yor
\a yór
\u H- yor
\ge s'arrêter
\rx v
\dt 12/May/2013
```

Parse Lexicon Structure

The ELAN-CorpA *parse lexicon* is used to speed up the *interlinearize* process as it contains already parsed and annotated words. If *parse lexicon* is loaded, the user can choose to search there first rather than launching the parsing process directly. *Parse lexicon* is an XML file that is linked to ELAN through its associated *.pfsx* parameter file, e.g.

```
<pref key="Interlinear.ParseSource">
    <String>C:\myProject\myELAN\parseBej.eafp</String>
</pref>
```

Here is its DTD:

```
<!ELEMENT parseLexicon (parse*)        >
<!ATTLIST parseLexicon
    dt    CDATA       #REQUIRED        >
<!ELEMENT parse (WForm, morph)         >
<!ELEMENT WForm (#PCDATA)              >
<!ELEMENT morph (Segment+)             >
<!ELEMENT Segment (#PCDATA)            >
<!ATTLIST Segment
    ge CDATA          #IMPLIED
    rx CDATA          #IMPLIED         >
```

the **<parseLexicon>** element is composed of parsed words <parse> and has a date of creation attribute *dt*. Each **<parse>** element has:

- ◊ a word form element **<WForm>** which contains the value of the word;
- ◊ a morpheme element **<morph>** containing the list of morphemes **** that compose the word. The latter contains the value of the morpheme, and has the attributes ***ge*** 'gloss' and ***rx*** 'category'.

```
<parseLexicon dt="12/May/2013">
<parse>
```

```
    <WForm>ago:jt</WForm>
    <morph>
      <Segment ge="1SG-" rx="PNG-">a-</Segment>
      <Segment ge="be_incapable\INT.PFV" rx="der.V1">go□j</Segment>
      <Segment ge="-COORD" rx="-CONJ">-t</Segment>
    </morph>
  </parse>
  <parse>
    <WForm>ahagit</WForm>
    <morph>
      <Segment ge="1SG-" rx="PNG-">a-</Segment>
      <Segment ge="wait\PFV" rx="V1">hagit</Segment>
    </morph>
  </parse>
  …
</parseLexicon>
```

The grid for the lexicon

The lexicon is displayed on a grid, with one or more lines per entry. The fields of the grid are:

- ◊ *Nr*: the line number;
- ◊ *Lexeme*: the value of the lexical entry;
- ◊ *Variant(s)*: alternate forms of the Lexeme, separated by a comma if there are more than one;
- ◊ *Gloss*: when an entry has different glosses, there is one line with the same Lexeme for each of them (for this reason the line number (*Nr*) is not the same as the identifier number (*id*) of the lexical entry);
- ◊ *TierX*: the category corresponding to the gloss of the lexical entry;
- ◊ *Underlying-Form*: underlying Segments of a compound entry, separated by a colon.

Sorting

Clicking any column header sorts the Lexicon grid by that field.

New entry

When a new entry is added to the lexicon, it appears at the end of the grid. The user may click on the *Lexeme* header to see that entry in its place in the lexicon grid.

Editing an entry

When an entry is edited and saved, it is deleted from the lexicon, and saved as the most recent entry.

The segmentation area

This area has two functions. Firstly, a grid displays potential morphological analyses for the target word, with one analysis per line and one morpheme per column. Secondly, when an analysis has been selected, and the selected segmentation morphemes have been validated in the lexicon, a grid shows the glosses. The cells in the grids respond to three *MouseEvents*:

- ◊ **Left-click**: filters the lexicon in the first grid to show all the entries concerned by the *segmentation* the cell belongs to;
- ◊ **Double-click**: selects the segmentation, opens the second grid and filters the lexicon to show the entries corresponding to the *value of the segment*;
- ◊ **Right-click**: opens a pop-up menu with a choice between *Insert a Record* or *Insert a Variant* into the lexicon, with the value of the cell as the entry (that can be edited).

Ergonomics of the morpheme glossing in the segmentation area

When a morpheme is selected in the segmentation grid, double-clicking on the corresponding item in the *lexicon* copies the *Gloss* and *TierX* values in the cells under the morpheme in the segmentation grid. Then the next morpheme in the segmentation area is selected (see Figure 4).

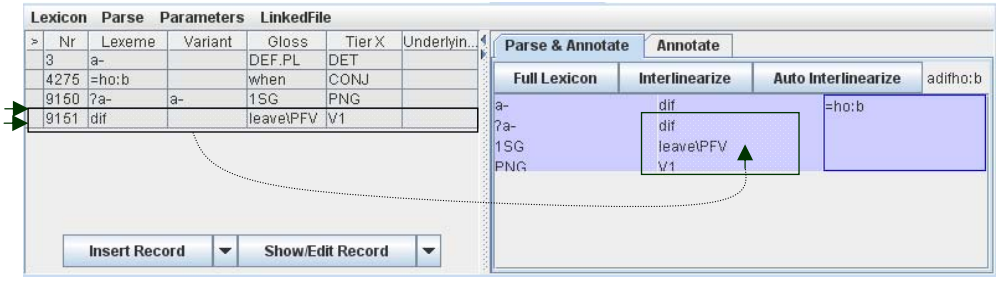

Figure 4: Annotation process

If the morpheme to be glossed is the last item in the segmentation area, selecting its gloss in the *lexicon* with a double-click, launches the process that copies the glossing annotations of the morphemes *from the segmentation area* to underneath the active word *in the annotation area*.

The process of creating the annotations under the word to be interlinearized

Once the parser has given a set of possible morphological segmentations for the target word, the user selects the correct one (by double-clicking on one of its morphemes) and chooses the correct annotations from the lexicon. Now these annotations have to be created under the word in the annotation area. Since this action modifies the annotation file, there is an option to return to a previous stage if necessary. The process of creating new annotations:

- ◊ deletes the existing annotations (i.e. children of the word being interlinearized) with the ELAN undoable command *ELANCommandFactory.DELETE_MULTIPLE_ANNOS*;

- creates the *mb* annotations under the current word, using the ELAN undoable commands: *ELANCommandFactory.NEW_ANNOTATION_VAL* for the first child, then *ELANCommandFactory.NEW_ANNOTATION_ VAL _AFTER* for the next ones;
- creates the *ge* and *rx* annotations using the same undoable command: *ELANCommandFactory.NEW_ANNOTATION_VAL*.

When this process is finished, it:

- selects the next word in the annotation area;
- removes the filter from the lexicon and clears the segmentation area.

Interlinearize Menu

The *interlinearize* process in ELAN-CorpA requires opening or creating a lexicon, then identifying which tier to start on and which tiers to place the annotations on. This is the subject of the *Interlinearize menu* which contains:

Lexicon

- **Create**: creates a new lexicon with a .eafl extension;
- **Open**: locates and opens an ELAN lexicon, linking it to the ELAN file;
- **Import**: imports a Toolbox lexicon by displaying a frame with two windows letting the user match the Toolbox fields to the ELAN lexicon fields;
- **Export**: exports the ELAN lexicon in the Toolbox standard format. Instead of a \ps field, there is an \rx field after each \ge field (gloss);
- **Save**: saves the ELAN lexicon, since the ELAN *file.save* command (ctrl/s) does not save the Lexicon files because these are linked. When closing ELAN, if the lexicon has not already been saved, a save prompt appears before quitting.

Parse

A file containing the segmentations and annotations of the words of the current ELAN file can be created or merged with an existing one. This is called a *Parse Lexicon* (cf. *Parse Lexicon Structure*). A Parse Lexicon can be opened and linked to the ELAN file in order to speed up the *interlinearize* process.

When a parse file is linked to the current ELAN file, a checkbox lets the *auto-interlinearization* process search in that *Parse* lexicon for the target word, and if that word has already been analysed (and therefore already exists in the parse lexicon), it copies the segmentations and annotations of the word directly, without asking, then moves on to the next word. It only stops if it finds an unknown word, or if a word appears in the lexicon with different annotations (homonyms in the parse lexicon). Then it asks the user to make a choice.

The parse menu contains three items:

- **Export Parse Data**: exports the segmentation and annotations of the words of the current ELAN file in a parse lexicon with the extension *.eafp*. The entries of the parse lexicon are unique except if the segmentation or gloss of a word has been analysed differently in the source file. In such cases there will be homonyms in the parse lexicon. The wordForms added to the parse lexicon are those of the current *interlinear tier* selected in the Parameters. So for example, if the ELAN file contains a dialogue, an *export & merge* should be done to merge the segmented and annotated wordForms of the second speaker (after having changed the *Parameters*) with those of the first;

- ◊ **Open Parse Data**: locates and opens a parse lexicon, and links it to the current ELAN file, so that it will open automatically next time the ELAN file is opened;
- ◊ **Export & merge Parse Data**: adds the segmentations and annotations of the words of the current ELAN file to an existing one. This selection will open a directory window to locate and select an existing parse file.

Parameters *(Setting up the Interlinearize or Annotation process)*

Interlinearize is a dual process involving both morphological segmentation of a word and glossing of the morphemes. The *annotation* process has been isolated to allow simple annotation of the elements on a tier by using any ELAN-CorpA lexicon. When the *annotation* tab is used, the parser is by-passed. It is possible to add new pairs of annotation tiers (with *ge* and *rx* types) to a source tier by *annotating* this tier with a different lexicon.

The *interlinearize* and *annotation* processes can be launched on any tier, so it follows that the user must make a choice. By default, the names of the added tiers are *mb (morpheme breaks)*, *ge (glosses)*, and *rx (categories)* but these can be changed.

The *Parameters* menu allows the configuration of each process. Three sub-menus are available to choose between creating new annotation tiers or using existing ones, and to define characters used to identify affixes in the lexicon.

◆ *Interlinearize Tier Parameter*: for setting up the *Interlinearize* process i.e. which is the starting tier and which tiers will receive the segmentation and annotations.

- ◊ **Configure Interlinearize tiers**: displays the *Configure tiers* window to **create new tiers** for interlinearization by choosing the segmentation tier from a dropdown list and editing the tier names to be created for annotation (*mb, ge* and *rx* by default). If the new tier name already exists, it is created again with a –cp extension;
- ◊ **Rename Interlinearize tiers**: displays the *Rename tiers* window to change the current tier configuration for interlinearization.

For these settings to be permanent when closing and re-opening an ELAN file, the parameters are saved in the *preferences* file (*.pfsx*) associated to the ELAN file. The keys for these parameters are: *Interlinear.Tier.Word, Interlinear.Tier.Parse*, *Interlinear.Tier.Gloss* and *Interlinear.Tier.Pos* in the following syntax:

```
<pref key="Interlinear.Tier.Word">
    <String>mot@SP</String>
</pref>
```

◆ *Annotation Tier parameter* lets the user set up the *Annotation* process (for which there is no parsing process):

- ◊ **Configure Annotation tiers** displays the *Configure tiers* window to let the user choose the annotation tier from a dropdown list and edit the name of the tier to be created (*ge* and *rx* by default). If the new tier name already exists, it is created again with a –cp extension.
- ◊ **Rename Annotate tiers** displays the *Rename tiers* window with the default names (*mb, ge* and *rx*), and allows the user to edit them. These settings are saved in the *.pfsx* parameters file in the following elements: *Annotate.Tier.Parse, Annotate.Tier.Gloss and Annotate.Tier.Pos.*

Morpheme break characters: to set up the special characters used in the lexicon to mark prefixes and suffixes (-), clitics (=) and stems (_). These delimiters can be reproduced on the subsequent tiers during the interlinearize process by checking the box *transmit morpheme break*

characters to annotation tiers. This parameter is saved in the *.pfsx* parameters file as 1 if checked, and as 0 otherwise:

```
<pref key="MorphemeBreakCharacter.Transmit">
    <Int>1</Int>
</pref>
```

Note that, if the *.pfsx* file associated to an ELAN file is lost, the interlinearize setup will have to be re-done.

LinkedFiles

When an ELAN-CorpA Lexicon or a Parse Lexicon is opened to annotate an ELAN file, these files are linked to it, so they will open next time the ELAN file is opened. This menu shows the filenames of the *Lexicon* and *ParseLexicon* currently open. By unchecking a filename, the associated lexicon will be unlinked to the current ELAN file (i.e. it deletes the *Interlinear.LexiconSource* and/or the *Interlinear.ParseSource* key(s) from the *.pfsx* file.

Various interfaces to Create or Edit a lexicon entry

During segmentation

The *interlinearize* process is interactive. The ELAN parser offers possible segmentations of a word with respect to the contents of the lexicon (words and affixes). When a segmentation is incomplete, that is to say a sequence remains that is not in the lexicon, it displays this sequence preceded by an asterisk in the segmentation area. By right-clicking on this segment, one can

insert the segment (the asterisk disappears) in the lexicon as a Lexeme or a Variant.

From the lexicon area

The **Insert Record** button: this button is reversible. It can be changed from *Insert Lexeme* to *Insert Variant* by means of a dropdown list.

From the lexicon grid

Once a line is selected in the lexicon grid area, a right-click on it lets the user choose *show associated record* which opens a show/edit window for this entry.

The Insert Lexicon data window

This window has two tabs (see Figure 5):

◊ **Insert Record** is for inserting a new Lexeme with a Gloss, a possible derivation function and a category (*TierX*). The gloss, derivation and category can all be multiple for the same entry;

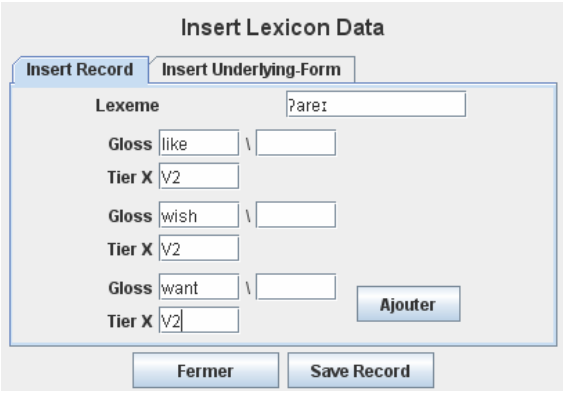

Figure 5: Insert Lexicon Data

◊ **Insert Underlying Form** is for entering an underlying form composed of other entries from the lexicon.

The Insert Variant window

This window has two tabs (see Figure 6):

◊ **Insert Variant** is for inserting an alternate form and selecting the lexicon entry it refers to from a dropdown list. A derivational function can be added in a separate cell which appears at the end of the morpheme and its gloss when it is annotated;

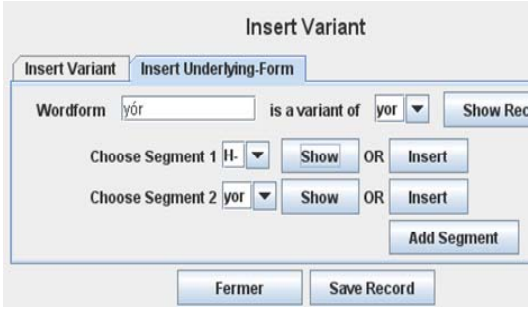

Figure 6: Insert Variant

◊ **Insert Underlying Form** is for entering an underlying form composed of other entries from the lexicon.

The Show/Edit window

This window has four tabs (see Figure 7):

◊ **Show Record** shows the contents of the chosen Lexeme;

◊ **Edit/Delete Record** allows the user to edit the entry value, delete variants of the entry and edit or delete the Lexeme contents;

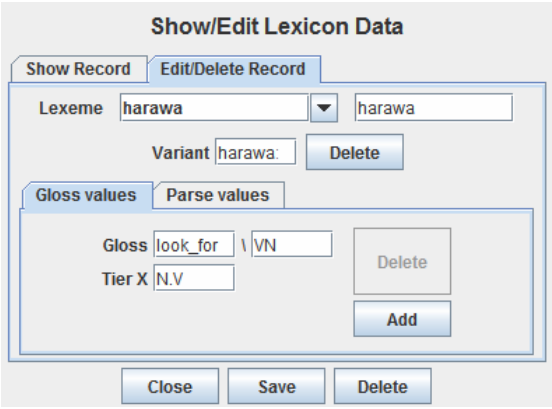

Figure 7: Show/Edit Lexicon Data

◊ In the ***Gloss values***, glosses, derivations and categories can be edited, added or deleted;

◊ In the ***Parse values***, segments can be changed or deleted.

The entry itself can be deleted by clicking the *Delete* button. The two other buttons let the user quit or save.

Conclusion

The new *interlinearize* function included in this version of ELAN-CorpA facilitates the morphological segmentation and annotation work of the field linguist, and above all ensures a certain consistency across the entire corpus. When a text has been annotated with ELAN-CorpA, the resulting file is the same as if it had been annotated by hand or imported from Toolbox. The structure of the ELAN file is not affected. Only the *.pfsx* parameter file (maintained by ELAN for each file) is different as it contains keys other than those in the Max Planck Institute's current release version. These keys let ELAN-CorpA know which lexicons are to be used, where they are located, and what the actual setup of the interlinearize or annotation process is for this file (word and morpheme breaks, glosses, categories and tier names). These keys do not prevent ELAN from opening the file correctly, they just will be ignored.

Unfortunately, the interlinearize tool cannot currently be added as a plugin to ELAN, though we would eventually like to develop this. In the meantime, each new release of ELAN requires us to add our own package to its sources and compile it as our own version of ELAN-CorpA. The main objective of the development reported in this article was to have a tool that would simulate the Toolbox interlinearize process to help researchers in their task.

We are confident that we have achieved this goal. Looking ahead, we will be working towards greater flexibility, so as to further expand the audience for this indispensable tool.