Linguistic documents synchronizing sound and text

Michel Jacobson, Boyd Michailovsky, John B. Lowe

Linguistic documents synchronizing sound and text

Michel Jacobson, Boyd Michailovsky, John B. Lowe, LACITO/CNRS, Paris

Abstract

The goal of the LACITO linguistic archive project is to conserve and to make available for research recorded and transcribed oral traditions and other linguistic materials in (mainly) unwritten languages, giving simultaneous access to sound recordings and text annotation.

The project uses simple, TEI-inspired XML markup for the kinds of annotation traditionally used in field linguistics. Transcriptions are segmented at the levels of, roughly, the sentence and the word, and annotation associated with different levels: metadata at the text level, free translation at the sentence level, interlinear glosses at the word level, etc. Time alignment is at the sentence (and optionally the word) level.

To minimize in-house development and maintenance, the project uses standard software to the extent possible. Marked-up data is processed using widely-available XML/XSL/XQL software tools, and displayed using standard browsers. The project has developed (1) an authoring tool, SoundIndex, to facilitate time-alignment, (2) a Java applet which enables standard browsers to access time-aligned speech, (3) XSL stylesheets which determine "views" on the data, and (4) a simple CGI interface permitting the user to choose documents and views and to enter queries. The paper describes these elements in detail. Current objectives are further development of the annotation with a view to linguistic research beyond simple browsing, and of a querying system (using a standard XML query processor) to exploit the annotated material.

Introduction

The purpose of the LACITO Archive project is the archiving of hundreds of hours of taped linguistic and ethnographic speech data, mainly in little-known and endangered languages, collected over the years by members of the CNRS (Centre National de la Recherche Scientifique) research group Langues et Civilisations à Tradition Orale (LACITO) in Paris. Apart from conservation, a major goal of the project is to make the recordings and, simultaneously, their textual documentation available to researchers, including those who recorded them in the first place, using modern, computer-aided research methods. The present paper describes a system, using freely available software, for the creation of documents giving simultaneous access to sound and text, which can be browsed and interrogated over a network.

The project first began in 1992 with the idea of digitizing original tapes and cassettes and conserving the recorded speech and other data on CD-ROM. Although this remains an important aspect of the project, it has become a routine technical operation and will not be discussed further here. The usual digital audio (CD-A) parameters of 44,1 KHz sampling rate and 16 bit samples have been adopted for digitizing and for archiving the sound data, although not necessarily for its dissemination.

The LACITO archive at present comprises a hundred or so texts in a score of languages, mainly languages of New Caledonia and of Nepal. The texts range in length from a few sentences (of the order of a minute in time) to several hundred (over half an hour), the total time being well over ten hours. Some of this material will be made available on the Internet in the near future. Over five hours of texts in New Caledonia languages have been produced for the multimedia library of the Tjibaou Cultural Center in Nouméa, New Caledonia, where they are available for consultation.

From the outset, importance was given to archiving the annotation (construed broadly to include transcriptions, translations, notes, etc., following Bird and Liberman 1999) which accompanies the recordings. Readers familiar with linguistic fieldwork will not need to be convinced that such annotation, and in particular the transcription, in large part worked out in the field in consultation either with the original speakers or with other members of the same speech community, is just as irreplaceable as the recordings themselves and requires far greater effort to prepare. One of the attractions of digitization is the possibility of archiving both recorded speech and annotation together on the same digital media, eliminating the possibility of their becoming separated and losing much of their value, a field linguist's recurrent nightmare. More positively, with appropriate tools, it facilitates the confrontation of the annotation with the recorded speech, and thus the correction and the enrichment of the annotation.

In the following pages we describe the ongoing development of a digital format associating speech recordings with annotation. We first briefly describe the format in which the annotation of such recordings has traditionally been published. Some of the LACITO materials had already been published in this form, without sound recordings; the rest remained in notebooks or unpublished files. An initial objective of the project was to archive this "legacy data", associating it with the recorded sound, without necessarily requiring further analysis. This required the development of a digital format for the project documents, coding the textual data and anchoring it to the sound resources. The documents are coded using the industry-standard XML (Extended Markup Language). We then describe methods of exploiting the documents (browsing, searching, etc.), giving simultaneous access to both sound and text. Because all of the project materials are coded as valid XML documents, we

are able to use standard, freely available software tools (XML parsers, query languages, stylesheets, and standard browsers) to exploit them, with a minimum of in-house development.

Encoding the documents

Structure of the legacy data

The structure of the traditional annotation, which will be familiar to linguists, is implicit in its typographical format (see fig. 1). The most important annotation is the transcription of the original language. Typically, this transcription is divided into segments corresponding roughly to intonation groups or to "sentences", and the sentences are further segmented into space-delimited "words". The sentences are numbered for reference; each one may or may not always begin on a new line. For our present purposes we will ignore other levels of structure: blocks of transcribed text may be divided into paragraphs, marked typographically by indentation, or into speaker turns; prefixes, suffixes, clitics, etc., included in typographical words may be marked off by separators (e.g. hyphens in fig. 1) and glossed separately.

1. nakpu nonotso siŋ   pa   laʔnatshe-m   are
   two   sisters   wood   make   go:REFL:3du–ASS   REP

2. siŋ   pa   lat-noŋ ban-noŋ   ban-noŋ   bilʊ   ʊxtotshe-m   are
   wood make   go-LOC forest-LOC forest-LOC tiger meet:3du–ASS   REP

3. bilʊ   ʊt-noŋ   oho   nono   nono–ai   nono–ai
   tiger meet-LOC   oh   elder.sister elder.sister–VOC elder.sister–VOC

   ine   ko   keːtsetso-roktsetso no rahecha   ni
   here TOP game   is   INFER   EMPH

1. They say that two sisters went to fetch wood.
2. When they went to fetch wood, they found a tiger in the forest.
3. When they found the tiger, [the younger sister said], "Why, big sister, there is game here!"

Fig. 1. Transcription, interlinear gloss, and free "sentence" translation

Two types of translation are commonly supplied, aligned (at least logically) with the transcription at different levels: free translations at the sentence level and glosses at the word level or below. The free translations may be printed after each sentence of the transcription, or, to facilitate connected reading, they may be grouped together at the bottom of each page or at the end of the text, in which case the translation of each sentence is co-indexed with the transcription. Where word-glosses are supplied, these are placed interlinearly and aligned under the corresponding words of the transcription. The glosses aid in the linguistic study of the text by indicating the semantic contribution of each word. There is no translation at a higher level than the sentence, but the free, sentence-level translations, strung together and numbered in the original order, serve as an intelligible, if not always smooth or elegant, translation of the whole text. The literal word-glosses obviously do not have this property: stringing them together in the order of the source language can be expected to yield gibberish in the target language.

To recapitulate, the annotation of fig. 1 is structured as a heirarchy of three levels, the "text", the "sentence" and the "word", with each category of annotation -- transcription, translation, or gloss -- associated with one level. Metadata concerning whole texts will be associated at the text level (see below). The structure is hierarchical in that a text contains sentences and a sentence words; it can be represented as a single-rooted tree. At the same time, the annotation has linear structure in that the transcription units at each level are sequentially ordered. This linear structure will be put into correspondence with the temporal structure of the recorded sound below.

Structured text

In the preceeding section, the structure of the most usual printed format for the annotation of speech recording has been derived from its typography. In this section we will outline a format in which the logical structure of this annotation is made entirely explicit. For example, a gloss will be identified explicitly by delimiting tags containing a label of our choice -- <GLS> at the beginning and </GLS> at the end -- and not implicitly by its position on the page, font, style, etc. It will be grouped (again in a labeled structure) into a superordinate element with the transcribed word of which it is a gloss. Text "marked up" in this way with labels identifying the functional, rather than the typographical, nature of each element is "logically structured text".

Structured text requires markup to label the logical nature of each element. In conceiving the archiving project, it was decided to use the widely-accepted norm for the definition of markup languages, SGML (Standard Generalized Markup Language), normalized as ISO 8879:1986. By this choice it was hoped to take advantage of the investment in SGML by the computer industry and by the academic Text Encoding Initiative (TEI), which had proposed SGML markup for a wide variety of document types used in literary and linguistic studies (Sperberg-McQueen and Burnard 1994). As the project was getting started, the World Wide Web Consortium adopted the XML (Extensible Markup Language) dialect of SGML as a recommendation (Bray 1998), triggering widespread development of free and public domain software, a phenomenon that had previously occurred around the display language HTML but never around the original SGML standard. The archiving project (with a number of other academic projects, including a renewed TEI consortium) adopted XML markup.

An XML document consists of structural markup and text data. It is organized into elements whose beginnings and ends are identified by user-defined structural tags. Between the start-tag and the end-tag, an element may contain other elements or textual data or both, e.g., a gloss 'dog' might be marked up as <GLS>dog</GLS>. In addition to tags, the structural markup may include attribute-value pairs. For example the language of the gloss 'dog' might be indicated by an attribute as <GLS lang="English">. Provision is made for reference to external resources, which may contain non-XML data such as sound or images. As mentioned, an element may contain other elements, but any contained element must be entirely nested in the containing element; there can be no overlap. The structure of an XML document is thus equivalent to a single-rooted tree.

The structural properties of an XML document or of a class of such documents can be declared in a Document Type Declaration (DTD). The DTD declares the elements that a document of the particular type may contain and the contents of each, defining the XML tree.

XML document-processing software begins by parsing the document to verify its "well-formedness", that is, its conformity with the syntactic rules of XML. A "validating

parser" may be used to verify not only the syntactic well-formedness of a document, but also its conformity with a particular DTD.

Markup of the legacy data as structured text

The LACITO project has loosely followed the TEI guidelines in designing the markup described here; more detailed markup will be adopted as the project evolves. The current encoding is sufficient to illustrate the use of XML and the integration of sound and text.

In the project's XML equivalent of the traditional structure described above, the textual data is contained (not always directly) in elements labeled TEXT, S, and W, corresponding to the "text", "sentence" and "word" levels of structure. The TEXT element contains a HEAD (see below) and a BODY. The BODY is made up of S elements. Each S element is uniquely identified (via the "id" attribute) and contains the transcription (TRANSCR), made up of W elements, and a free translation (TRADUC). There may be more than one free translation, with the target language indicated by the value of an attribute. Each W element contains a FORM element -- the transcribed word -- and a GLS element -- the word-gloss. The words of the free translation are not explicitly marked up since there does not seem to be any reason to access them individually. The HEAD contains metadata concerning the whole text, such as its title, the date, the speaker, the language, etc. Fig. 2 shows the markup of a fragment of a text in Hayu, a Tibeto-burman language spoken in Nepal:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE TEXT SYSTEM "Archives.dtd">

<TEXT id="hayu1" lang="hayu">
<HEAD>
    <TITLE>Two sisters</TITLE>
    <SOUNDFILE src="SOEURS_SOUND"/>
</HEAD>
<BODY>
...
<S id="hayu1s1">
    <TRANSCR>
        <W><FORM>nakpu</FORM><GLS>two</GLS></W>
        <W><FORM>nonotso</FORM><GLS>sisters</GLS></W>
        <W><FORM>si&#331;</FORM><GLS>wood</GLS></W>
        <W><FORM>pa</FORM><GLS>make</GLS></W>
        <W><FORM>la&#660;natshe-m</FORM>
            <GLS>go:REFL:3du-ASS</GLS></W>
        <W><FORM>are</FORM><GLS>REP</GLS></W>
        <PUNCT>.</PUNCT>
    </TRANSCR>
    <TRADUC lang="Francais">On raconte que deux soeurs allèrent
    chercher du bois.</TRADUC>
    <TRADUC lang="Anglais">They say that two sisters went to fetch
    wood.</TRADUC>
</S>
...
</BODY>
</TEXT>
```
Fig. 2: Markup of a linguistic text

Time-alignment markup

Sound recordings, by nature, can be segmented as finely or as coarsely as desired, certainly more finely than required to detect any linguistically significant element. The main

decision to be made in aligning sound and text is the length of the smallest segments of sound to be indexed and made individually accessible. Since the documents in question here are typically presented as examples of connected text, it was decided to index segments of connected text rather than, say, words or phonemes. On the other hand, because the languages of the texts would be unfamiliar to most users, it was desirable to keep the segments short. The simplest choice was to index the S-units.

To preserve access to stretches of S's, or to the recording of the whole text without interruption, the original recordings are not segmented into separate S-sized sound resources but kept whole, with the segments corresponding to each S identified by their starting and ending time-offsets as measured from the beginning of the sound resource. The consequences of this -- in our view inescapable -- decision are of practical significance because XML (like other languages of the SGML family) provides for only global references to external, non-XML resources. Accessing a part of a sound file is beyond the capacity of standard processing software and has required some in-house software development, as will be seen below.

An XML document can reference external "entities", which may consist of either "parsable" (i.e. XML formatted) or "non-parsable" (non-XML) data. The XML standard does not indicate how the internal structure of a non-XML entity can be accessed. The entity declaration, in this case of the non-parsable sound resource, specifies an entity name (used in future references), a physical location (computer file name and path), identification of the contents as "non-parsable" (that is, non-XML), and identification of the data-type (WAV): `<!ENTITY SOEURS_SOUND SYSTEM "Hayu/SOEURS.wav" NDATA wav>`. The declaration may be located in the XML document which calls the entity in question, but we have chosen to centralize all such declarations in the project DTD. This makes it possible to manage the archive sound resources globally (moving them or switching between compressed and uncompressed versions) without modification of the individual text documents, which use unchanging entity names.

It would equally be possible to simply specify the name of the soundfile and the path in the XML document (for example: `<SOUNDFILE href="Hayu/SOEURS.wav"/>`), but such declarations could not be centralized, and they only implicitly indicate the nature of the contained data. (It is true that XML parsers cannot at present use the latter information.)

For alignment data we have adopted a solution similar to that suggested by the TEI (P3 11.2.5), incorporating the start- and end-time offsets as attributes, not (as in the TEI proposal) of the structural element that they refer to (in our case the S) but of an AUDIO element included in that element. The following shows how the AUDIO element would be added to the markup of (1).

```
<S id=" hayu1s1">
    <AUDIO start="2.3656" end="7.9256"/>
    ...
</S>
```
Fig. 3: The AUDIO element.

(The oblique at the end of the empty AUDIO element obviates the need for an end-tag.) We will discuss below how these indications are interpreted.

As mentioned above, the sound recording has a linear structure which can be measured either by time or sample count, both external to XML. Typically, the start-offset of each S coincides with the end-offset of the preceding S, but this is not necessary, and overlapping is possible. Thus, in a conversation, speaker B might start before speaker A has finished:

```
<S who="A">
    <AUDIO start="0.00" end="5.00"/>
    <TRANSCR>I haven't finished talking.</TRANSCR>
</S>
<S who="B">
    <AUDIO start="4.00" end="9.00"/>
    <TRANSCR>And I have already started.</TRANSCR>
</S>
```

Fig. 4: Speaker overlap reflected in sound indices.

The XML structure does not reflect the overlap, and indeed XML would not allow partial overlap between the beginning of <S who="B"> and the end of <S who="A">. (The TEI (P3 11.3.2) has proposed a notation to cover this situation, and to show precisely where in the transcription such overlap begins and ends.) Nevertheless, the temporal overlap in the recorded sound can be indicated as in the example, because the values of these attributes are not XML structural elements and are not verified by the XML parser.

The project DTD allows time-offsets to be specified for each W as well as for each S:

```
<S id=" hayu1s1">
    <AUDIO start="2.3656" end="7.9256"/>
    <TRANSCR>
    <W><FORM>nakpu</FORM><AUDIO start="2.3656" end="3.0500"/></W>
    <W><FORM>nonotso</FORM><AUDIO start="3.0500" end="4.0000"/></W>
    ...
    </TRANSCR>
    <TRADUC lang="Anglais">They say that two sisters went to fetch
        wood.</TRADUC>
</S>
```

Fig. 5: Multi-level time-alignment.

In this case, the XML markup parallels the temporal structure. Once again, the parser will not verify that W time-offsets are nested between the start-offset and end-offset of the containing S.

We have used the fact that an XML parser cannot detect overlapping time offsets above to allow overlapping speech turns. However, such permissiveness can go too far, and it may be necessary to develop software to check for anomalies in time anchoring for different sorts of documents. For example, we might wish to verify the following:

that the start offset of any unit has a lesser value than the end offset.

that start and end offsets of hierarchically inferior units are included within those of the containing hierarchically superior unit.

that the relation $end_n \leq start_{n+1}$ hold between offsets of successive elements of the same type. This requirement would be relaxed for overlapping speech turns or for documents in which the order of elements is not necessarily fixed, for example, a wordlist annotation keyed to a recording elicited in a different order.

Remarks on markup

The archiving project markup may be briefly situated among existing and proposed markups for the annotation of corpora of connected speech. Annotations intended for the study of the phonetic or acoustic aspects of language will not be considered, although these routinely include time-alignment. Coding systems like that used by the Summer Institute of Linguistics' well-developed and widely used SHOEBOX software (Buseman and Buseman 1998), or the CHILDES format (MacWhinney 1995), for which extensive software has been

developed, could accommodate the LACITO's legacy data, but they offer somewhat limited structuring possibilities as compared to SGML/XML, and they do not give direct access to newer standards and tools such as Unicode, web browsers, standard query languages, formatting languages, multimedia, etc.

The original TEI guidelines for orthographic speech transcription (Sperberg McQueen and Burnard 1994, Ch. 11) using SGML are extended in the Corpus Encoding Standard (CES, Ide and Priest-Dorman 1999) to cover morphosyntactic annotation comparable to the LACITO annotation. The CES is proposed as an exchange format for speech corpora. The MATE (Multilevel Annotation, Tools Engineering, Dybkjær et al. 1998) project, which is still at an early stage of development, proposes not only a common XML exhange format for a large number of existing corpora with very different codings and purposes, but also development of a software workbench of analysis tools. Time alignment with recorded data is provided for, with the alignment data contained in the primary "transcription document". The even more recent TalkBank (1999) promises to produce an XML annotation for transcribed speech, the "codon", based on the "annotation graph" model of Bird and Liberman (1999), and extensive software tools.

One (optional) feature of the CES architecture, later adopted by the MATE project, uses the TEI "multiple hierarchies" mechanism. Annotation is separated into "primary" or "original data" -- often an orthographic transcription -- kept in a "hub" document, and higher-level "analyses", kept in separate marked-up documents (one per analysis, with a separate DTD for each), multiply linked to the elements (e.g. words or sentences) of the hub document. The hub document, which contains the time anchoring data in the MATE model, is regarded as "unannotated" and unchanging, while the analyses may be multiple, perhaps contradictory (e.g. different syntactic analyses), with overlapping hierarchies not permissible in a single SGML or XML document. The concept of "primary data" requires some revision for field linguistics, where there is usually no conventional orthography, and where any transcription is "constructed data" which cannot be regarded as stable. However, a transcription may still be accepted as "basic", even if it is not unchanging, and the time-anchoring will usually be stable in any case. Our reason for not adopting this architecture is that our current (and immediately projected) annotations do not require it.

Software tools and implementation

Authoring tools

A major practical difficulty in preparing the archive documents is measuring the time-offsets required to align the transcriptions with the recordings and inserting them into the AUDIO element described above. Since this function is not available in standard document preparation software, software has been written for this purpose, associating a sound editor with a text editor.

The program SoundIndex plays a sound file (WAV, AIFF, or CD-Audio track) and simultaneously displays the waveform and a text, normally the transcription prepared by the linguist. It allows the user to mark on the waveform and record in a table the start- and endpoints corresponding to each segment of the transcription. The original version, written in C++ for the Macintosh platform, is not XML-aware, but accepts any text whose segments are delimited by a particular character (e.g. carriage return, full stop, or -- as in fig. 6 -- opening square bracket) and creates a text table of start and end time offsets (the window HAYU1.IDX in fig. 6). A Perl script is used to integrate the transcription, any other annotation, and the time offsets into an XML document conforming to the project DTD. This

version of SoundIndex, whose user interface screen is shown in fig. 6, is fully documented and freely available (see http://lacito.vjf.cnrs.fr/ARCHIVAG/ENGLISH.htm). It has been used by a half-dozen LACITO researchers, and we have been able to convert the resulting implicitly structured documents to time-aligned XML format. At least one user, in Australia, was able to use the program "off the shelf" and took the trouble to review it (Thieberger 1999).
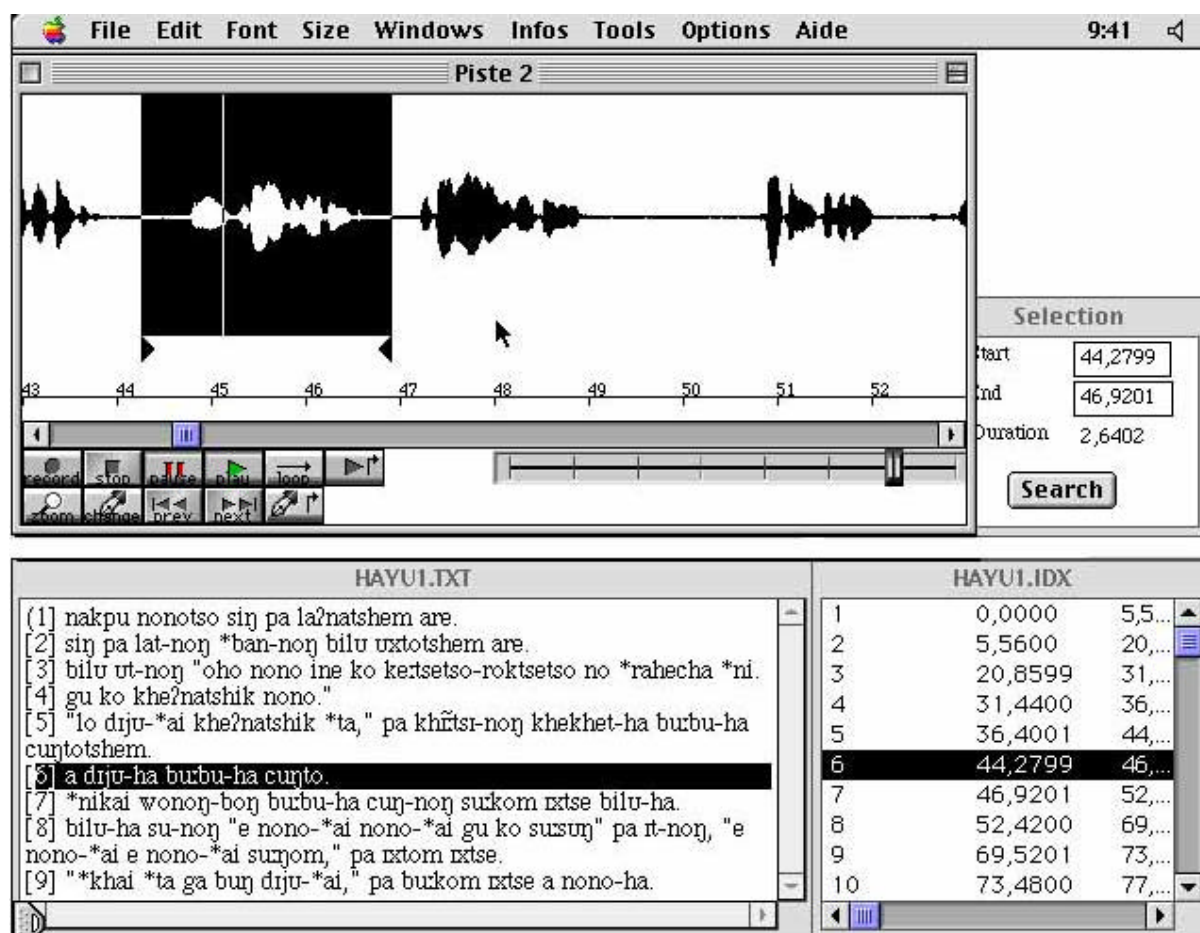

Fig. 6: SoundIndex v. 1

A new version of SoundIndex, developed recently in Tcl/Tk, works directly with XML documents. The user specifies the label of the elements to be time-aligned. When any such element is selected and the corresponding part of the waveform identified, the program generates an AUDIO element with the time-offsets and incorporates it into the element (fig. 7). This version was partly inspired by Claude Barras's Transcriber (Barras et al. 1998), in particular in the use of the Snack Tcl/Tk sound extension (Sjölander 1999). Transcriber also uses XML, providing an time-anchoring interface geared to the direct entry of transcriptions of radio news broadcasts into a template XML document of predefined type. One difference between the programs is that SoundIndex uses a standard XML parser to open documents, so that it can open any XML document and guarantee its well formedness. It reads the text from a standard DOM interface for display. A preliminary version of SoundIndex v. 2 is available on the project website.
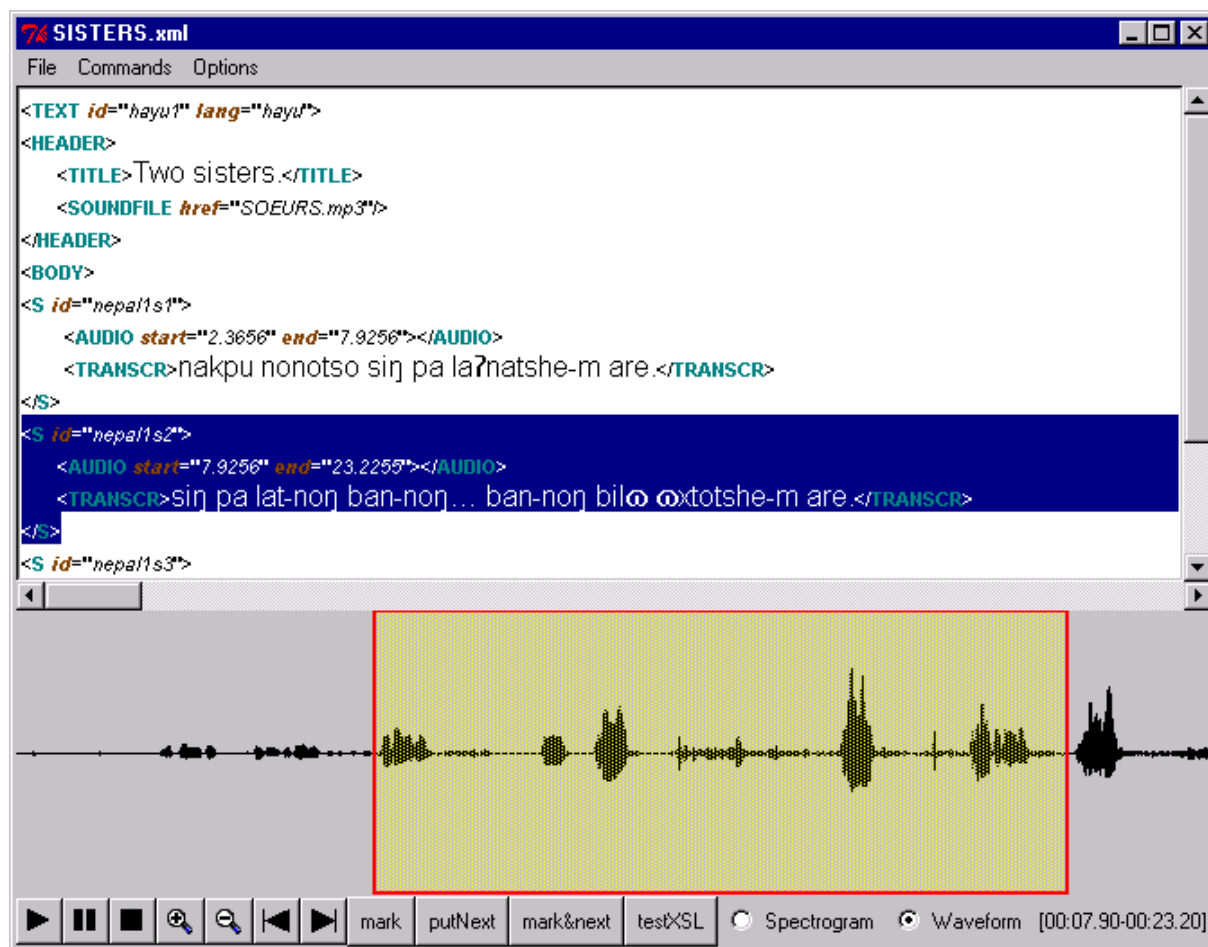
Fig. 7: SoundIndex v. 2

Although the process of time-alignment might seem fastidious, field linguist users generally appreciate the detailed access to their recorded data and profit from the time-alignment process to improve their transcriptions. This caused us to provide for editing in the text window of SoundIndex.

The project has so far been geared to the processing of legacy data and has not made extensive use of XML editors or authoring tools. XML documents have typically been generated from implicitly structured text by scripts, which also convert special characters into character references to Unicode codepoints. Corrections to the documents are made to the non-XML versions, and the XML markup is regenerated. Scripts for converting text to and from XML can be written in any programming language that provides a DOM (or SAX) interface to an XML parser and has the capacity to apply regular expressions, etc., to Unicode coded text. Both Perl and Tcl have been used as scripting languages, and Java for stabilised routines. SoundIndex v. 2 (on the model of Transcriber) is a first step in the direction of editing documents directly in XML; this will no doubt become the rule as tools become available and as we move away from legacy data.

In working with new texts, the most fastidious task for the linguist is the entry and systematization of word-glosses, where these are desired. The SIL's SHOEBOX program (Buseman and Buseman 1998) provides tools which control and aid text and gloss entry by reference to a lexicon, simultaneously enriching the lexicon based on the entered text. (It does not provide access to sound recordings.) It also can partly automatize the production of a morphological transcription. This kind of authoring tool for texts and lexicons will need to be

developed for XML-coded data. In the meanwhile, existing SIL "standard format" files can readily be converted to XML by script.

Browsing tools

To facilitate wide dissemination of the documents, it was decided to build the browsing system around CGI scripts and standard web browsers. In principle, the documents are accessible to any client running a standard browser and equipped with a soundcard and a Unicode font. Documents are dynamically translated into HTML on the server (this may become unnecessary as XML-aware browsers become available), with incorporated scripts to handle user input (e.g. mouse-clicks). Access to segments within the sound file, not provided for in HTML, is handled by a Java applet. A typical display screen is shown in fig. 8.
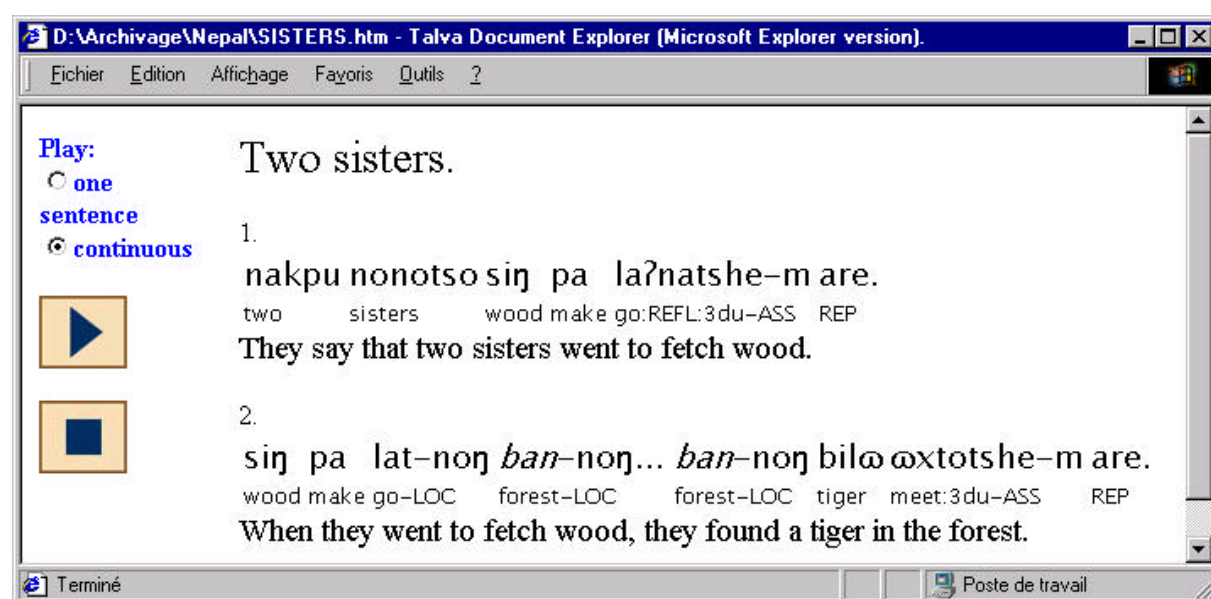

Fig. 8: Browsing of the archive via a browser

On the server, the archives are composed of XML files (data documents and XSL stylesheets) and sound files. Two types of XML-aware programs run on the server, (1) a query processor and (2) an XSL (see below) stylesheet processor which formats XML, including query responses, as (in the present case) HTML pages.

Browsing is based on the linkage between text and sound. The synchronization of text and sound is either text-driven, each point in the text linking to a continuous segment of the sound resource, or sound-driven, that is, driven by the internal temporal structure of the sound resource, each instant during the playing of the sound resource linking to an element of text (or to two or more overlapping elements, in the case of multi-level markup or overlapping speaker turns). The user chooses a segment of text, which is either played alone ("one sentence") or initiates the playing of the remainder of the file ("continuous"). In the latter case, the sound-driven mode is activated: as the sound is played the corresponding segments of text are highlighted.

Display format

The user may choose among a predefined number of "views" on the browsed document. For example, he may choose to see only the transcription, or the transcription and the free translation, or the transcription and the aligned word-glosses; he may also choose the

language of the translation if more than one is available. In fact, the user chooses among predefined stylesheets in XSL (Clark 1999, Deach 1999); these define the selection and the order in which data is displayed and format it into the display language (here HTML, but TeX, RTF, etc., can also be produced). The following is an extract from an XSL stylesheet:

```
<xsl:template match="BODY">
    <OL><xsl:apply-templates select="S"/></OL>
</xsl:template>
<xsl:template match="S">
    <LI>
    <xsl:apply-templates select="TRANSCR/W/FORM"/>
    <xsl:attribute name="ID">
        <xsl:value-of select="@id"/>
    </xsl:attribute>
    <xsl:attribute name="onclick">
        playFrom('<xsl:value-of select="@id"/>')
    </xsl:attribute>
    </LI>
</xsl:template>
```

Fig. 9: Part of an XSL stylesheet

The first "rule" instructs the processessor, when the BODY element is encountered, to create an HTML ordered list; it specifies that only S elements and their contents are to be considered. The second rule instructs it to create a new list item each time an S element is encountered, and to include in it only transcribed word-forms (TRANSCR/W/FORM). In addition, the processor stores the id attribute of the current S element. If the user clicks on the text of the S, a playing function (`playFrom(id)`) is called with the id attribute as an argument, and the corresponding segment of the sound resource is played.

Application of the stylesheet (5) to the sample document (1) produces the following HTML:

```
...
<ol>
    <li ID='hayu1s1' onclick="play('hayu1s1')">
        nakpu nonotso siŋ pa laʔnatshe-m are
    </li>
...
</ol>
```

Fig. 10: HTML resulting from application of the XSL style

Both the text display and the call to the playing applet are defined by XSL rules. On initialization, the applet is passed the name of the sound entity as well as the identifiers (`id`) of all of the anchored segments of the text and the start- and end-offsets corresponding to each.

```
<APPLET code="myAudioApplet.class" width="0" height="0" name="player">
    <PARAM NAME="file" VALUE="Sample1.wav"/>
    <PARAM NAME="IDS" VALUE="hayu1s1, hayu1s2, hayu1s3, hayu1s4"/>
    <PARAM NAME="STARTS" VALUE=" 2.3656, 7.9256, 23.2255, 33.8056"/>
    <PARAM NAME="ENDS" VALUE="7.9256, 23.2255, 33.8056, 38.7657"/>
</APPLET>
```

Fig. 11: Sample HTML-formatted declaration and initialization of the Java applet

Here the "player" applet is instructed to use the sound resource "sample1.wav". Since this is a sound (and not an image or video) resource, it is not necessary to reserve screen space (`width=0 height=0`). Four pairs of start- and end-offsets are passed to the applet,

corresponding to the HTML elements whose `id` attributes are "hayu1s1", etc. Given this information, the applet can either play the appropriate sound when passed an `id` attribute (text-driven mode) or, from the current time-offset during continuous play of the sound, look up the `id` attribute of the corresponding text segment and pass it to the script in the HTML file. In the latter case (time-offset or sound-driven mode) the HTML script displays and highlights the appropriate text as it is being played .
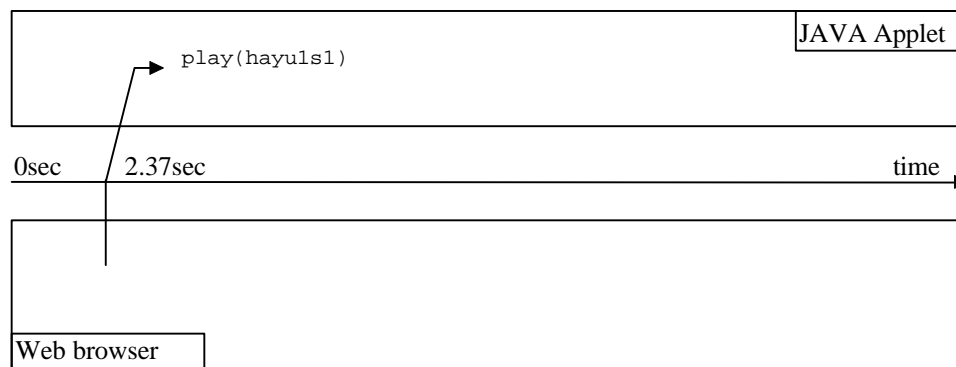
Text- and sound-driven synchronization

Text-driven synchronization is the most straightforward. The text consists of a series of elements which may or may not be anchored. Each of the anchored elements has an `id` attribute permitting its unambiguous identification and an `onclick` attribute which calls the Javascript function `play(id)`, with the current `id` attribute as an argument, when the user clicks anywhere inside the element (see (6) above). The Javascript function `play(id)` serves as an interface, passing the command `cmd_play(id)` to the Java applet which executes the sound resource. The applet looks up the corresponding start- and end-offsets and plays the sound, stopping when the end-offset is reached. The JavaScript function also executes the function `active(id)`, which highlights the corresponding text segment.

```
function play (id) {
    if (document.player.cmd_play(id)) {
        active(id);
    }
}
```

Fig. 12: The JavaScript function `play(id)`

Thus the user can choose to play a segment by clicking anywhere in the corresponding text. The communication occurs from the browser to the applet, as indicated in the schema below:

Fig. 13: Communication from script to applet in text-driven mode

Sound- (or time-offset) driven synchronization is used when the user asks for the entire sound resource to be played from a given point. As the sound is played, the anchored segments must be displayed by the browser. Three types of events are generated by the applet to make this possible: "start" and "end" events when a given segment is reached and completed, and a "stop" event when the end of the resource is reached. These events are independent of each other and are periodically checked for by the browser.

The following is a more detailed description of the process. When the user clicks on a location in the text, the JavaScript function `playFrom(id)` is activated and calls the applet, passing it the `id` argument of the text segment. The applet finds the corresponding entry in its

index list, and builds a stack containing all of the events that will have to be generated in executing the sound resource from that entry to the end of the resource. The applet then begins execution of the sound resource.

The navigator checks periodically for applet-generated events. In the example below, the `loop()` function is executed every 50 ms. Each time it is polled (by `cmd_NextEvent()`), the applet checks whether any new events need to be generated and, if so, calls the corresponding JavaScript functions, `startplay(id)` or `endplay(id)`, to advance the display. If execution of the resource is stopped, either because the end has been reached or because of a user interruption, the applet empties its stack and generates "end" events for all current segments and a "stop" event, calling the JavaScript function `stopplay()`. The script then stops checking for applet-generated events.

```
function startplay(id) {
    activate(document.all.item(id));
}
function endplay(id) {
    desactivate(document.all.item(id));
}
function stopplay() {
    clearTimeout(timer);
}
function playFrom (id) {
    if (document.player.cmd_playFrom(id)) {
        timer = setTimeout("loop()", 50);
    }
}
function loop() {
    document.player.cmd_NextEvent();
    timer = setTimeout("loop()", 50);
}
```
**Fig. 14: JavaScript functions used in time-driven mode**
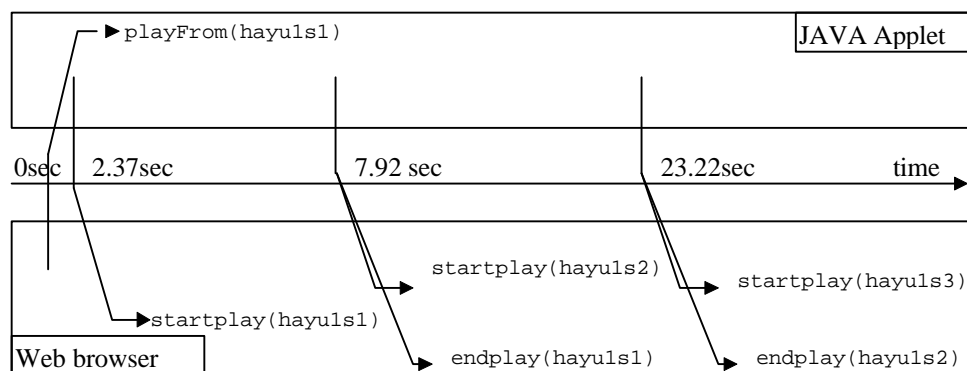
Fig. 15: Communication between script and applet in time-driven mode

Between the start and end events corresponding to a single segment, other events may occur, such as the beginning of another temporally overlapping or nested segment. The schema below shows the order of events generated by the applet under these conditions.

id2 follows id1	partial overlap	id2 nested in id1
startplay(id1)	startplay(id1)	startplay(id1)
endplay(id1)	startplay(id2)	startplay(id2)
startplay(id2)	endplay(id1)	endplay(id2)

```
        endplay(id2)              endplay(id2)              endplay(id1)
```
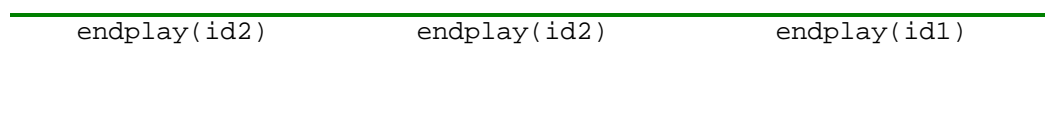
Fig. 16: Order of execution of anchored segments

Managing and querying the archive

A major incentive for the use of a structured text coding is the possibility of data processing beyond browsing, such as management and interrogation of the archive as a text database. We are experimentally using and testing some of the currently implemented XML query language processors: XQL (Robie 1998), XML-QL (Deutsch 1998), LT XML (Thompson 1997).

For managing the entire archive, a simple query with sggrep (a querying tool of LT XML) according to criteria coded in the document headers brings up the titles of all documents in a given language (ARCHIVES/TEXT[lang='Hayu']/HEAD/TITLE). The result of such a query, formatted by XSL and displayed, serves as the interface for choosing a document for browsing. A similar query could serve to dynamically constitute a corpus of texts for study and interrogation (ARCHIVES/TEXT[lang='Hayu']).

Others implementations of query languages such as XQL or XML-QL are more powerful and offer better compliance with Unicode. Performance is mediocre (in part limited by our server), so that we are limited to querying a single text at a time. However, the fact that field linguists generally work with relatively small corpora, and that XML query languages are still in the prototype stage, are cause for optimism.

In one prototype XQL querying application, a straightforward interface permits the user to enter a "word" or a regular expression and then, after passing the string to the query language and formatting the response, returns a KWIC (key word in context) listing of all S units in which the string occurs as a W/FORM. Because the response includes the sound resource declaration, and the S units are returned with their AUDIO elements, the user can click on any S in the KWIC listing and hear the sound. For this kind of application (and for linguistic analysis in general), some improvement in the annotation, in particular inclusion of a morphological transcription, subordinate to the word level and in parallel with the word transcription, will be necessary, since users will generally wish to specify either roots or affixes, not affixed or inflected word forms, in searches. The same requirement applies to accessing an online lexicon from text documents. Prototype XML-coded lexicons of Hayu and Limbu (Tibeto-Burman languages of Nepal) are currently under development.

Future development

The key to the LACITO archive project from the technical point of view has been the adherence to standards, XML in particular, which gives access to a large and rapidly developing inventory of tools maintained (elsewhere) to keep up with changing hardware, web standards, Unicode, etc. This environment makes it possible for a project with limited programming resources to concentrate on developing specifically linguistic tools. Since there are many such projects, we may expect that the distributed development of software for field linguistics and other specialities will become the rule.

Notes

The initial prototype documents were time-aligned using a Hypercard program and marked up in HTML by J.B. Lowe in 1996. Michel Jacobson wrote the current versions of

SoundIndex, and all other programs, XSL stylesheets, etc., used by the project. Our ideas on structured text have been influenced by Robert W. Hsu of the University of Hawaii, author (beginning in the early 1970s) of the LEXWARE system for structuring and processing digital dictionaries.

Sample documents and information about the LACITO Archive project may be found on the project website, http://lacito.vjf.cnrs.fr/ARCHIVAG/ENGLISH.htm.

The Archiving project received support under the Language Engineering program of the department of Human and Social Sciences (SHS) of the French National Center for Scientific Research (CNRS).

References

Adler, Sharon, et al. 2000. Extensible Stylesheet Language (XSL). W3C working draft. (http://www.w3.org/TR/WD-xsl)

Barras, Claude, Edouard Geoffrois, Zhibiao Wu and Mark Liberman. 1998. "Transcriber: a Free Tool for Segmenting, Labeling and Transcribing Speech, Proceedings of the First International Conference on Language Resources and Evaluation (LREC), Granada, Spain, pp. 1373-1376. (http://www.etca.fr/CTA/gip/Projets/Transcriber/Index.html)

Bird, Steven, and Mark Liberman. 1999. A formal framework for linguistic annotation. Technical Report MS-CIS-99-01, Computer and Information Science, University of Pennsylvania. (http://xxx.lanl.gov/abs/cs.CL/9903003)

Bray, Tim, Jean Paoli, C. M. Sperberg-McQueen, eds. 1998. Extensible Markup Language (XML) 1.0. W3C Recommendation. (http://www.w3.org/TR/REC-xml)

Buseman, Alan and Karen Buseman. 1998. The Linguists SHOEBOX. Summer Institute of Linguistics. (http://www.sil.org/computing/catalog/shoebox.html)

Clark, James, ed. 1999. XSL Transformations (XSLT) version 1.0. W3C Recommendation. (http://www.w3.org/TR/1999/REC-xslt-19991116)

Deutsch, Alin, Mary Fernandez, Daniela Florescu, Alon Levy, Dan Suciu, eds. 1998. XML-QL: A Query Language for XML. Submision to the W3C XSL Working Group. (http://www.w3.org/TR/1998/NOTE-xml-ql-19980819.html)

Dybkjær, Laila , Niels Ole Bernsen, Hans Dybkjær, David McKelvie and Andreas Mengel. 1998.The MATE Markup Framework. MATE Deliverable D1.2. (http://mate.nis.sdu.dk/information/d12/)

Ide, Nancy and Greg Priest-Dorman.1999. Corpus Encoding Standard - Document CES 1. Version 1.5. (http://www.cs.vassar.edu/CES/)

Robie, Jonathan, Joe Lapp, David Schach, eds. 1998. XML Query Language (XQL). Proposition to the W3C XSL Working Group. (http://www.w3.org/Style/XSL/Group/1998/09/XQL-proposal.html)

SIL. 1998. LinguaLinks. Version 3.0. Overview and installation guide. SIL. Dallas. (http://www.sil.org/lingualinks/LingTool.html)

Sjölander, Kåre. 1997-99. The Snack Sound Extension for Tcl/Tk (http://www.speech.kth.se/snack/)

Sperberg-McQueen, C. M. and Lou Burnard, ed. 1994. Guidelines for Electronic Text Encoding and Interchange. (TEI P3). Electronic Book Technologies. [revision date 16 May 1994. On CD-ROM. Citations are by section number.] (http://www-tei.uic.edu/orgs/tei/)

Thieberger, Nick. 1999. Using SoundIndex to transcribe fieldnotes. (http://www.linguistics.unimelb.edu.au/CALW/sindex.html) [We are grateful to this user for drawing our attention to a few bugs, since corrected (and not mentioned in the review). The 100-line limit mentioned in the review is in fact a limitation of the Mac Perl editor.]

Thompson, Henry, Richard Tobin, David McKelvie, Chris Brew. 1997. LT XML Software. (http://www.ltg.ed.ac.uk/software)

MacWhinney, B. 1995. The CHILDES system. W. Ritchie and T. Bhatia, eds. Handbook of Language Acquisition. New York. Academic Press. (http://www.childes.psy.cmu.edu).

Other web pages

Lacito Archive project: http://lacito.vjf.cnrs.fr/ARCHIVAG/ENGLISH.htm

Linguistic annotation: http://morph.ldc.upenn.edu/annotation/

TalkBank: http://www.talkbank.org