Recent Developments in the Emu Speech Database System

Lasse Bombien[1], Steve Cassidy[2], Jonathan Harrington[1], Tina John[1], Sallyanne Palethorpe[3]

1. Institut für Phonetik und Sprachliche Kommunikation, University of Munich, Germany.
2. Department of Computing, Macquarie University, Sydney.
3. Macquarie Centre for Cognitive Science, Macquarie University, Sydney.

Abstract

The Emu Speech Database system is a set of software tools developed to support research in acoustic phonetics and other corpus based speech research. This paper describes recent updates that have been made to the latest release of the software.

1. Introduction

The EMU Speech Database System, which has been developed over a number of years (e.g., McVeigh & Harrington 1992; Harrington, Cassidy, Fletcher and McVeigh, 1993; Harrington & So, 1994; Cassidy, 1999; Cassidy & Harrington, 1996, 2001; Harrington, Cassidy, John and Scheffers., 2003) is an integrated set of tools for creating, querying and analysing annotated speech corpora. In common with many speech software systems, it includes facilities for annotating speech data from synchronized waveform and spectrographic displays, as well as a library of routines for digital speech processing. It differs from these, however, in allowing hierarchical and autosegmental annotations and in providing a query language for extracting annotations from these structures and their associated signal files. Since 1988, when it was first developed as 'Acoustic Phonetics in S' at CSTR, Edinburgh University (Taylor, Caley, Black and King., 1999), EMU has provided a transparent interface to a powerful graphical and statistical programming language (first 'S', then 'Splus' and since the mid 1990s, the R programming environment - R Development Core Team, 2006). Another distinguishing feature of EMU is its organization of a speech database in a so-called *template file*, which specifies the physical location of the signal files and the types of hierarchical and autosegmental relationships between sets of labels. The template file was included to facilitate the sharing of speech corpora across different users, working at different sites, and possibly on different platforms. Another feature, which is unique to EMU, is the possibility of building various kinds of labeling structures automatically using the Tcl scripting language (Welch, Jones and Hobbs, 2003). More recently, an interface has been built to Praat, a computational system for doing phonetics (Boersma & Weenink, 2001), allowing Praat labeling structures to be converted into EMU and vice-versa (Harrington et al., 2003).

2. Current EMU structure

In the last three years, EMU 2.0 has been provided with a new internal structure (Fig. 1) that allows individual parts of the system to be modified and tested independently. Both the front-end and the back-end have been designed in a modular way and can be extended without much effort.

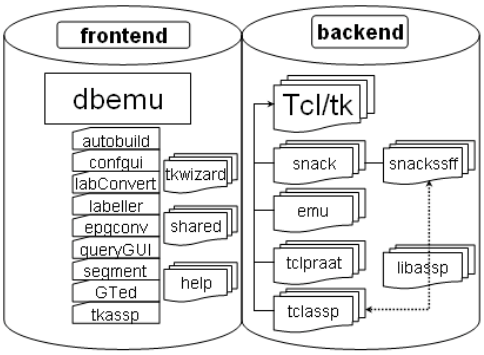

Figure 1: the new EMU 2.0 structure.

In earlier versions, the EMU back-end consisted of three parts providing:

(1) the database functionality,
(2) signal processing (trackdata)
(3) signal display (padgraph).

These parts were integrated into the Tcl interpreter differently depending on the platform, e.g., by loading shared libraries on Windows while *nix (all unix derivates) used custom interpreters with built-in functionality. Subsequently, Snack (Sjölander, 2000) was used for displaying waveform and spectrographic data. One of the advantages of Snack is that it is easily extendable to support, for example, the SSFF file format frequently used in EMU; another is that it provides the facility for zoomable spectrograms. Accordingly, the new

EMU back-end consists of the platform independent Tcl extensions, emucore (database functionality) and snack.
Two more extensions are planned as part of the back-end:

(1) In order to facilitate the use of Praat from within EMU, an extension tclpraat has been developed. This extension is used to send commands to Praat via sendpraat (Boersma and Weenink, 2006)

(2) So far, the emu application *tkassp* for signal processing has made use of the assp tools (advanced speech signal processing) by invoking subprocesses. As the assp tools are currently being rewritten as a library supporting 64 bit systems, a new extension *tclassp* will be used in future rather than calling subprocesses. The assp library will also be used to add support for the SSFF file format in snack.

The EMU front-end has undergone significant changes, too. While in earlier releases, all applications had to be invoked separately, EMU 2.0 includes a front-end *dbemu* (see Fig. 2) which interfaces with all current EMU applications and provides an overview of all accessible databases and their utterances. *dbemu* constitutes the new core of all applications while the former core, the *EMU labeller*, is now one application among others in the EMU system. All EMU applications are separate Tcl packages which can share their functionality. While all of the applications can be launched from within *dbemu*, most of them can still be started on their own.

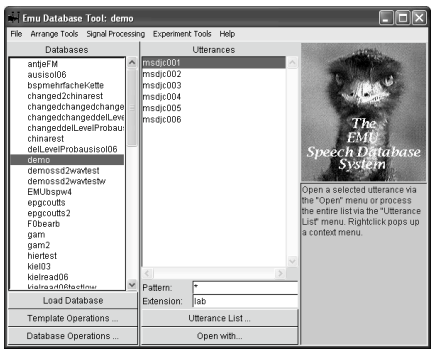

Figure 2: the new EMU 2.0 frontend – the dbemu interface.

A major problem for collaborative work between groups of researchers on the same database was in how to store the template file and the speech data and annotations. For EMU 2.0, template files can be stored anywhere that the computer has access to. Thus, neither the speech data and annotations nor the template file needs to be stored on the machine that runs the EMU system. The various paths to the template files are accessible and can be defined using a new gui *emu-conf*, while the location of the different databases is viewable directly from the *dbemu* database list. The new internal structure of the EMU system removes the limitations of the earlier system on the size of signals that can be annotated with EMU.

From *dbemu* there is an interface to a gui *GTemplateEditor* for creating template files in which the user fills in information about the location of the signal and annotation files of the database, provides information about the type of annotation structure, and specifies both the types of display that should appear when the database is accessed as well as any Tcl scripts for building annotation structures. After a database template has been written, it shows up as the corresponding database name in *dbemu*.

3. Signal Processing

While in other speech processing systems, parametric data such as formants and pitch are often calculated on the fly, EMU makes use of the assp tools or their graphical interface *tkassp* for calculating derived signal files. The assp tools include routines for filtering, and for pitch, formant, and spectral analysis. A library with these functions is currently in development to replace the present command line programs. *tkassp* can be launched either in standalone mode, or else *dbemu* can be used to pass a list of utterances from the database to *tkassp* for signal processing. *tkassp* offers multiple types of input selection: single files, directories, file and directory lists, EMU utterance lists, and EMU segment lists. The output files can be stored by either specifying a single output directory but also by maintaining the directory structure of the input. The latter is specifically useful for complex database structures. The output files of *tkassp* can easily be added to the database as additional tracks using the database template. There is also a tcl-script for conversion of raw EPG3 (electropalatographic) files to the SSFF format that can be interpreted by EMU. Finally, EMU allows both formant and fundamental frequency tracks to be manually corrected with a mouse and updated.

4. Annotation

Loading the database causes all that database's utterances to appear. These utterances can be opened by the *EMU labeller* tool, in which annotation is supported by the Tcl/Tk programming language (Welch et al., 2003), or else they can be opened by the other integrated tools *wavesurfer* and *praat*.

The *EMU labeller* supports an annotation model that allows hierarchical relations between labels. EMU distinguishes timeless annotation levels from time-bounded Event and Segment annotation levels. All levels can have hierarchical relations to each other; this enables rich annotation structures (Cassidy and Harrington, 2001; Harrington et al., 2003).

Further development will introduce new annotation views that are more suitable to larger scale annotations that are needed, for example, in dialogue analysis, while retaining the same ability to integrate detailed phonetic or prosodic annotations into a hierarchical structure.

All annotation levels, their hierarchical relationships, view options and relationship to time are defined in the database template using a graphical template editor (*GTemplateEditor*). The template editor also allows legal labels to be specified as well as labels to be decomposed into features.

In previous versions, it was possible to edit EMU annotations only using the *EMU labeller*, but in this most recent version, all time-dependent levels can be edited in *Praat* (Boersma and Weenink, 2001) and in *WaveSurfer* (Sjölander & Beskow, 2005) without losing the hierarchical structure of the EMU annotation. While

WaveSurfer innately supports the label format produced by ESPS, Waves+ and EMU, for *Praat* the label files are converted to textgrids (and back) on the fly. Label levels that are not directly anchored in time, i.e. that inherit their times from other labels, are necessarily disregarded when editing in *Praat*/*WaveSurfer* (since they can only deal with time-bound labels). However, on quitting *Praat*/*WaveSurfer*, the EMU hierarchy will be updated and where necessary rearranged.

The EMU system has provision for including custom Tcl scripts for automatic annotation (AutoBuild-Scripts) or more complex modules by appropriate definitions in the database template. Some predefined scripts are integrated in the EMU system as well as two complete modules: *epgdisplay* for palatogram visualization and *hier2sigview* for displaying timeless annotation levels in the signal view of the *EMU labeler*.

AutoBuild-Scripts are used to automatically build hierarchical relations between annotation levels using temporal alignment, dictionaries etc. A new graphical tool, the *AutoBuild wizard*, facilitates an easy deployment of AutoBuild-Scripts over an entire database or selected parts of it, eliminating the need to load each utterance for this task. Any AutoBuild-Script can be used from within the wizard, that is, not necessarily the one defined in the database template.

5. Querying the database

The EMU query language (Cassidy & Bird, 2000) enables researchers to isolate speech segments based on both sequential and hierarchical context. There can be simple queries such as 'return all segments from the Phonetic level' but also complex queries like 'return all segments from the Phonetic level that precede a vowel nasal sequence in intermediate phrase-initial words following the definite article'.

The *EMU 2.0 Query Tool* provides a graphical query tool *queryGUI* that assists the user by creating a query string, thus removing the need for the user to learn the syntax and semantics of the query language itself. With a few simple mouse clicks, a query string can be created and sent to the *EMU Query Tool* that runs the queries over the chosen utterances of the database.

Each query results in an utterance list where all query matching segments are listed with their onset and offset times and the utterance they belong to. For these segments, signals can be extracted from within the *EMU Query Tool*. All the results can be saved to files for further analysis. The EMU Query Tool and thus the *queryGUI* are accessible from the *dbemu* interface.

6. Analysis- R statistical Environment

The EMU system provides a package within the *R programming language* for statistical computing. *R* is similar to the S language and environment which was developed at Bell Laboratories (formerly AT&T, now Lucent Technologies) by John Chambers and colleagues. The *EMU-R package* enhances R by including the facility for querying annotations and extracting signal data from EMU databases (see Harrington, forthcoming, for numerous examples of formant, spectral, and EPG analyses in R). Querying can be done either directly from within *R* or by loading stored output files of the graphical query tool. Query results and signal data can be further manipulated, displayed and statistically evaluated in *R*. Additional functions for plotting and signal processing are also included. In future, it will be possible to make use of *R*'s Tcl/Tk interface, in order to launch (graphical) EMU tools from within *R*.

7. Conclusions

EMU 2.0 provides a user-friendly access to all EMU functions using graphical user interfaces, a new internal structure in the programming code, and the integration of other third part tools *Praat* and *Wavesurfer*. EMU runs on Windows and UNIX platforms and can be downloaded from the EMU sourceforge website (http://emu.sourceforge.net) together with some databases (see also Harrington, forthcoming). Current development includes completing the port to the Macintosh OS X platform.

8. References

Boersma, P. and Weenink D. (2001), *Praat, a system for doing phonetics by computer*, Tech. Report 132, Inst. Phonetic Sciences, Univ. Amsterdam. Retrieved on 2006-11-21 from http://www.praat.org/

Boersma, P. and Weenink D. (2006), *Sendpraat: sending messages to a Praat shell program* (Code Version February17, 2006) [Computer Program], Retrieved on 2006-11-21 from http://www.fon.hum.uva.nl/praat/sendpraat.html.

Cassidy, S. (1999), Compiling multi-tiered speech databases into the relational model: experiments with the EMU system. In *Proc. Eurospeech*, 2238-2242. Budapest, Hungary.

Cassidy, S. and Bird S. (2000), Querying databases of annotated speech. In *Proc. Eleventh Australasian Database Conference*, 22, 12-20. Canberra, Australia.

Cassidy, S. and Harrington, J. (1996). EMU: an enhanced hierarchical speech database management system. In *Proc. Sixth Australian International Conference on Speech Science and Technology*, 361-366. Adelaide, Australia.

Cassidy, S. and J. Harrington (2001), Multi-level annotation in the Emu speech database management system, *Speech Communication*, 33, 61-77.

Harrington, J. (forthcoming). *The Phonetic Analysis of Speech Corpora.* Blackwell. Retrieved on 2006-11-21 from http://www.phonetik.uni-muenchen.de/~jmh/research/emupapers/pasc.htm

Harrington J., Cassidy S., Fletcher J. and McVeigh A. (1993), The mu+ system for corpusbased speech research, *Computer Speech and Language*, 7, 305-331.

Harrington, J., Cassidy, S., John, T. and Scheffers, M. (2003). Building an interface between EMU and Praat: a modular approach to speech database analysis. *International Congress of Phonetic Sciences*. Barcelona, Spain.

Harrington, J. & So, L. (1994). Some design criteria in segmenting and labelling a database of spoken Cantonese. In *Proc. Fifth Australian International*

Conference on Speech Science and Technology. 215-220. Perth, Australia.

McVeigh, A. and Harrington, J. (1992). The mu+ system for speech database analysis. In *Proc. Fourth Australian International Conference on Speech Science and Technology*. 548-553. Brisbane, Australia

R Development Core Team (2006). *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing.

Sjölander, K. (2000), *The Snack sound extension for Tcl/Tk*, Retrieved on 2006-11-21 from http://www.speech.kth.se/snack/

Sjölander, K. and Beskow, J. (2005). *WaveSurfer - an open source speech tool,* Retrieved on 2006-11-21 from http://www.speech.kth.se/wavesurfer/

Taylor, P., Caley R., Black A. W., and King S. (1999). *Edinburgh Speech Tools Library, System Documentation Edition1.2*. CSTR, University of Edinburgh.

Welch, B. B., Jones K., and Hobbs J. (2003). *Practical Programming in Tcl and Tk* (4 ed.) Englewood Cliffs, N.J.: Prentice Hall.