ELSEVIER

Speech Communication 33 (2001) 61–77

SPEECH COMMUNICATION

www.elsevier.nl/locate/specom

Multi-level annotation in the Emu speech database management system

Steve Cassidy [a,*], Jonathan Harrington [a,b]

[a] *Speech Hearing and Language Research Centre, Macquarie University, Sydney, NSW 2109, Australia*

[b] *Macquarie Centre for Cognitive Science, Macquarie University, Sydney, Australia*

Accepted 2 August 2000

Abstract

Researchers in various fields, from acoustic phonetics to child language development, rely on digitised collections of spoken language data as raw material for research. Access to this data had, in the past, been provided in an ad-hoc manner with labelling standards and software tools developed to serve only one or two projects. A few attempts have been made at providing generalised access to speech corpora but none of these have gained widespread popularity. The Emu system, described here, is a general purpose speech database management system which supports complex multi-level annotations. Emu can read a number of popular label and data file formats and supports overlaying additional annotation with inter-token relations on existing time-aligned label files. Emu provides a graphical labelling tool which can be extended to provide special purpose displays. The software is easily extended via the Tcl/Tk scripting language which can be used, for example, to manipulate annotations and build graphical tools for database creation. This paper discusses the design of the Emu system, giving a detailed description of the annotation structures that it supports. It is argued that these structures are sufficiently general to allow Emu to read potentially any time-aligned linguistic annotation. © 2001 Elsevier Science B.V. All rights reserved.

Keywords: Speech databases; Speech annotation

1. Speech database systems

In the last ten years, speech and language researchers have made increasing use of shared spoken and written corpora, and recent surveys show that well over a hundred different corpora worldwide are now available (for example by the LDC [http://www.ldc.upenn.edu/] and ELRA [http://www.icp.grenet.fr/ELRA]). The increased use of corpora can be attributed to a number of factors including the increased capacity of computer systems to store and manage several gigabytes of data, the need in speech and language technology research for training and testing materials, and the desire to avoid replicating similar speech and language materials in many different laboratories, especially since creating usable corpora is very demanding on resources. The increase in corpus development has also been driven by a greater emphasis on modelling spontaneous speech as an alternative to research based on citation-form data from a small number of talkers usually of the same gender and accent which are still typical of many experiments in speech science and experimental phonetics.

[*] Corresponding author.

Although large corpora are founded on the principle of shared data and resources, this goal is often unachievable either because the corpora are delivered without any tools to search and analyse them, or else because they presuppose that laboratories have access to similar software systems for speech and language analysis that were used to create them. Over the last ten years, we have developed the Emu speech database system (Harrington et al., 1993; Cassidy and Harrington, 1996) in order to begin to address the problem of providing a set of tools for interrogating and analysing speech corpora; more recently these tools have been extended to include modules for speech database creation. An overarching goal in building this system has been to avoid prescribing a methodology for corpus development, both because we consider this an entirely unrealistic aim, and also because it would limit, rather than enhance, the range of speech and language corpora (and segmentation and labelling methodologies) that researchers consider appropriate for the many different kinds of tasks to which corpora are applied. We therefore aim instead to develop a set of software tools for use in database research that are both flexible and that make very few prior assumptions about the design criteria of the speech and language corpora themselves.

2. Review

A number of earlier projects have addressed the issue of multi-level annotation of recorded speech data and have provided software tools for creating and querying these annotations. Many of these projects (for example, some of the projects described in an earlier special edition of this journal: Kurematsu et al., 1990; Hedelin and Huber, 1990; Carlson et al., 1990; Hendriks, 1990) have defined annotation structures including multiple levels of annotation. It is difficult to determine from the published descriptions of these projects whether they were ever used for more than one speech database project. Their annotation structures are typically tailored to one kind of enquiry and it is not clear whether the number or names of the annotation levels could be changed in any way. The existence of these systems illustrates the need for a general purpose speech database system which can support arbitrary structures within its annotation scheme.

A more recent attempt at providing a flexible multi-level annotation scheme is the Partitur project from the Bavarian Archive for Speech Signals (Schiel et al., 1998). Partitur builds upon the SAM label file format used in various European speech projects and defines a set of annotation levels with predefined meanings. Links are defined between some of these levels to indicate, for example, the inclusion of words within a phrase or speaker turn or the association between an orthographic word and a canonical pronunciation for that word. Although the file format and user tools associated with Partitur are capable of handling arbitrary annotations and level names, the official Partitur format prescribes a set of level names and their interpretations. It is clear that Partitur is both a file format/annotation formalism and an annotation standard; and it is this combination of providing both a file format and annotation formalism that makes it easier to exchange data between different projects. Partitur emphasises the simple SAM file format which is amenable to searches and modifications using simple Unix text file manipulation tools such as `awk` or `grep`; since the users of Partitur are familiar with the use of these tools, no special purpose annotation creation or search tools are needed.

One of the few systems which attempts to provide a solution to the problem of using differently formatted speech databases is the QuickSig system (Altosaar et al., 1999). QuickSig is implemented in Common Lisp on Macintosh systems and defines a mapping between the original annotation in a database and a standardised internal annotation format based on interconnected Lisp objects. Facilities are provided to map various machine readable phonetic alphabets onto Worldbet (Hieronymus, 1994) to enable cross-database queries. The major problems with QuickSig lie in various implementation issues; for example, the entire database must reside in memory while it is being used which obviously limits the size of database that can be dealt with. In order to import a new database into the system, LISP code must be written to interpret the annotation file formats and map them to the internal QuickSig model. Nevertheless, QuickSig provides a rich annotation model and the use of the LISP environment allows powerful user level tools to be built on top of the database application.

Query languages proposed for speech database systems have tended to accompany particular speech database projects and software systems and hence reflect the internal structure of annotations in these systems. Almost every speech database collection project which has used a simple flat file format for storing labels has produced software tools for searching these files in various ways. A good example is the *findsegs* tool distributed with the ShATR multi-microphone corpus (Crawford et al., 1994). ShATR is annotated at multiple levels of detail and the labels are stored in collections of simple label files. The *findsegs* tool allows the different levels of annotation to be searched for segments matching various criteria which can include overlapping segments and surrounding context. Once segments are found their start and end times are reported so that the corresponding speech data can be retrieved. The authors do not intend this tool to be applicable to anything other than this database and it provides a query language tailored to the study of multiple talker, multiple microphone recordings.

In summary, in the past there have been almost as many speech database systems as there have been speech databases. The small number of systems which attempted to provide general purpose facilities have not gained widespread use. There is a clear need for general purpose standards and tools to support the creation, management and analysis of annotated speech databases.

3. An overview of Emu

The Emu speech database system has been in use for a number of years and a distinct pattern of use has developed in that time. The structure of the Emu software reflects this pattern of use and so it is instructive to outline it here. As Fig. 1 shows, the Emu system works with annotated speech data in a number of formats; the speech data may have been marked up using the Emu toolkit or may have been obtained elsewhere. A common practice is to use the ESPS/Waves+ toolkit from Entropic [http://www.entropic.com] or the Emu labeller to produce initial markup for a database and then extend this annotation using Emu scripts written in Tcl. Once the annotation is complete, Emu is used to query the database for tokens matching various criteria; the result of such a query is a list of the tokens which match the criteria along with their start and end times. This list can be used to extract any of the time aligned data associated with the annotation. Both query results and extracted data are usually read into analysis

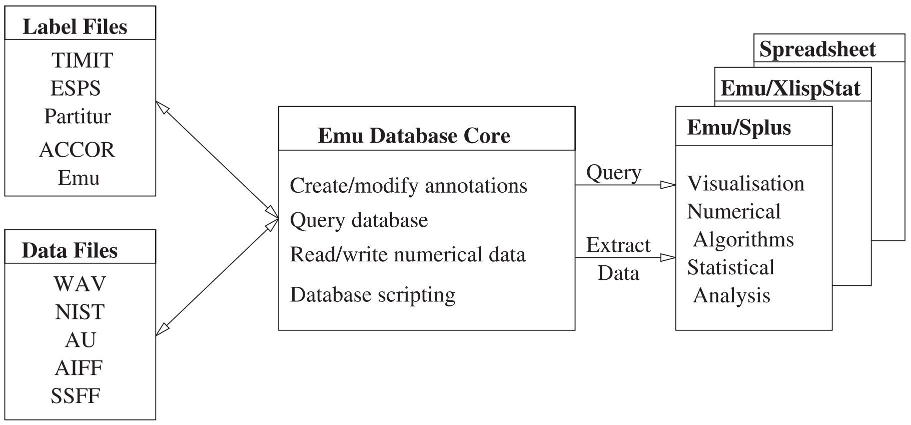

Fig. 1. The flow of data in the Emu speech database system. The Emu database core provides a number of interfaces for database manipulation, including a graphical labeller.

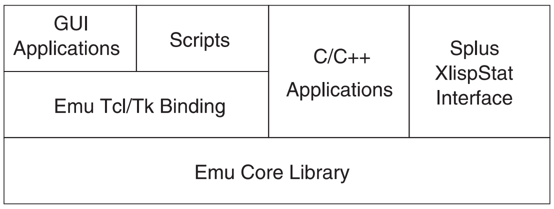

Fig. 2. Layers within the Emu toolkit. The core C++ library implements a uniform interface to speech data and annotations and is built upon by applications written in various languages.

packages such as Splus or Xlisp-Stat, where various routines have been written to facilitate analysis and visualisation of speech data.

3.1. Implementation

The Emu system (Fig. 2) is built around a core library, written in C++, which provides primitive operations on annotations such as adding and removing tokens, defining relations between tokens and performing queries on one or a set of complete annotations. This core library acts as the interface between the Emu user or programmer and the various annotation formats understood by Emu; programs can be written to manipulate databases irrespective of their format.

The primary operations implemented by the core library are reading and writing annotations and numerical data in various formats; adding, deleting and modifying tokens in an annotation; and searching a database for tokens matching constraints on labels and annotation structure.

These operations are implemented in a platform [1] and database independent manner. The user and programmer need not be concerned with issues of data and label file formats, but instead should be able to use a natural set of operations to construct and manipulate annotations.

Input and output of annotations and time series data are a central part of the functionality of the Emu system. Our goal is to enable Emu to be used with any file format which fits the constraints of the internal representations used in the system. A number of file formats are dealt with in the current version of Emu [2] but importantly, a programming interface is provided for adding new label and data file formats to the system. In each case, routines must be added for reading annotations or data into the internal structures used by Emu. Once these routines are present, the new formats can be used transparently by the whole Emu system.

The core functionality is provided both as a C++ library and as a set of extensions for the Tcl (Ousterhout, 1994) scripting language (Fig. 2). Emu extends the Tcl language with commands to manipulate annotations and special purpose user interface elements for graphical displays of speech data, including spectrograms. These are used in the standard Emu user tools but are also available for use in new applications.

3.2. Data extraction and analysis

The Emu query engine serves to select a number of tokens from the database according to various criteria such as their labels and location within the annotation structure. The result of a query is a table including start and end times which can then be used to extract numerical data corresponding to the tokens which matched the query. Each utterance in an Emu database can have any number of time aligned data tracks associated with it. The Emu system provides facilities for extracting the data corresponding to every matching token from a query, and delivering this in some form to analysis routines written in either C++, Tcl, Splus or Xlisp-Stat.

As illustrated in Fig. 1, the endpoint of the data derived from a speech database by the Emu system is usually a statistical environment such as Splus [http://www.mathsoft.com/splus] or Xlisp-Stat [http://www.stat.umn.edu/~luke/xls/xlsinfo]. Data from an Emu database imported into these environments is stored in a special data structure

[1] Emu currently runs on various Unix systems and Microsoft Windows 9x/NT. A port to the Macintosh is in progress.

[2] Currently Emu reads label files from Waves+, TIMIT, SpeechStation and the ACCOR project. Emu reads sampled speech data in the WAV and NIST Sphere formats on all platforms and can use the Edinburgh Speech Tools library (Taylor et al., 1998) on Unix systems to read many more formats.

which separates the data for each token but allows operations on all tokens as a group. Routines are provided which operate on this kind of data structure, applying a common operation to all tokens or drawing various kinds of graphs to summarise the data. For example, it is easy to extract all the formant data for a set of vowels and plot averaged, formant trajectories for each vowel overlaid on the same diagram. An earlier paper on the MU+ system (Harrington et al., 1993) described many of the special purpose graphics and analysis routines built on the Splus environment; all this functionality is retained in the current implementation.

The same level of access to numerical data is provided via the C++ library and can be used in cases where costly numerical processing is to be carried out for each token. As an example, the Emu source distribution contains a contributed program which extracts the data for each token returned from a query and performs multiple Fourier transforms which are then averaged and output for further analysis (Watson, 1999).

More details of the programming API and the facilities provided by the user tools can be found in the Emu manual which can be downloaded from our web site [http://www.shlrc.mq.edu.au/emu/].

3.3. The Emu labeller

The primary interface for creating annotations within the Emu system is the graphical Emu Labeller (Fig. 3). The labeller presents the annotation time aligned with views of the sampled speech signal and associated data tracks in a similar manner to many other speech annotation systems. It is also able to show a graphical representation of the domination or association structure of an utterance which can be edited manually if required. The labeller is intended to be an extensible annotation platform which can be adapted to new applications via plugins which can, for example,

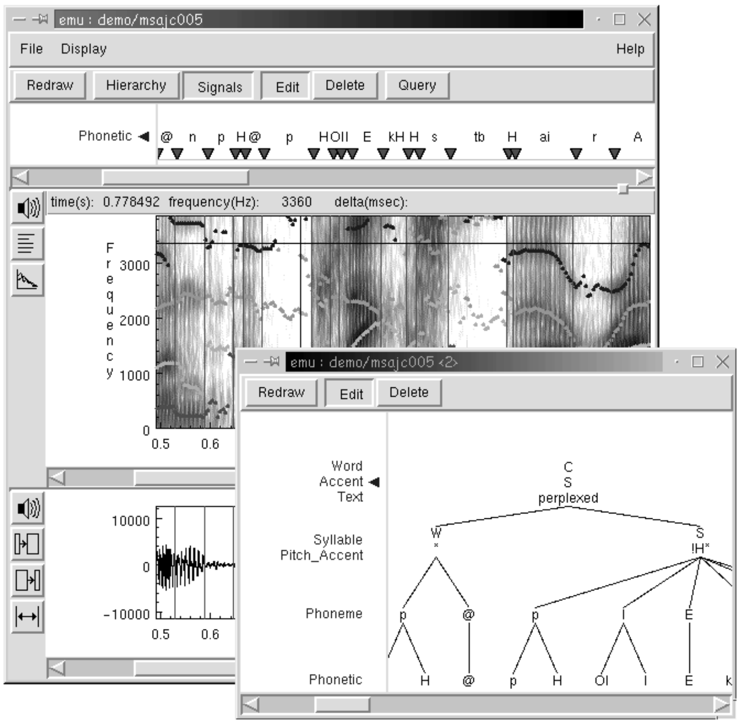

Fig. 3. The Emu Labeller showing two views of the annotation. The time-aligned Phonetic annotation is displayed along with various signals. The hierarchical annotation is displayed separately and is not time-aligned.

display special kinds of data or automate parts of the annotation process.

In its *Signal View* (Fig. 3, large window), the labeller can display any time series data as an *x–y* graph including multi-track data such as formant frequencies. The displays can be zoomed and scrolled and the system attempts to ensure a sensible alignment between the labels and the signal displays. Spectrograms can be overlaid with formant frequencies if available; [3] other special purpose displays can be provided via the plugin interface. The top part of the display shows the annotation levels which are defined to have explicitly timed segments or events. The boundary times of segments and the times of events are shown as small triangles and the token labels are shown associated with these times. Tokens can be created or edited by direct manipulation of the display using the mouse and keyboard.

The *Hierarchy View* of the annotation structure (Fig. 3, smaller window) shows the relations between tokens at different levels. The user can select which set of levels is displayed and the labeller renders a view which shows the relations that exist between each pair of neighbouring levels. No time information is used to generate this display, and tokens are laid out so as to produce a reasonable representation of the relational structure of the annotation. In this view, new tokens can be added at non-timed levels only and links can be made between tokens using the mouse. The hierarchy view can be displayed either in the main window, or in a separate window so that it can be viewed and edited at the same time as the signal view.

New functionality can be added to the labeller via the plugin interface. Plugins are segments of code written in Tcl/Tk and optionally C/C++ which can provide arbitrary extensions to the labeller. An important use of plugins is to provide special kinds of data display; for example, a plugin has been written to display electropalatographic (EPG) data aligned with the current cursor position. The plugin facility might also be used in the future to allow display of video data. A simpler interface is provided for the special case of automatically building all or part of an annotation.

[3] Emu does not include a formant tracker or any other signal processing tools. Spectrograms will be calculated from the raw speech signal but formants and other derived parameters must be generated with third party software such as the Edinburgh Speech Tools (Taylor et al., 1998) or Waves+ [http://www.entropic.com].

4. Annotation structure and system design

4.1. Annotation structure

An Emu database consists of a number of *utterances* corresponding to some convenient unit of analysis such as a word or a sentence. Each utterance consists of one or more *levels* of annotation: each level contains zero or more *tokens* which may or may not be associated with time information.

Each *level* in an Emu annotation contains tokens which denote a qualitatively different kind of linguistic information. A level has a theoretical interpretation and a corresponding set of criteria for deciding what constitutes a token at that level.

A *token* corresponds to a single linguistic object which has one or more labels as defined by its level in the annotation, and optionally has associated time information. The tokens within a level are stored as a partial sequential ordering, meaning that there may be sequence relations between pairs of tokens, but that each token may have zero, one or many following tokens. The ordering is defined either by the start times of the tokens, or explicitly by the creator of the annotation.

Since there are no constraints on the times recorded for individual tokens within a level, gaps and overlaps are readily accommodated. Emu distinguishes two kinds of tokens: *events*, which are associated with a single instant in time, and *segments*, which are associated with a start and an end time. Each level within an annotation is declared to be made up of either segments or events. Within a segmented level, tokens may have gaps or overlaps as is appropriate for the application.

Events and segments are differentiated because they enter into different kinds of relationships. Events can never be ‘parents’ in a domination relationship since domination implies temporal inclusion which in turn implies a discrete time span.

Hence declaring a level to be made up of events allows some validation of an annotation structure to be carried out. There are many kinds of linguistic objects which are best represented as occurring at an instant rather than across a span of time, such as pitch-accents in a ToBI prosodic annotation (Beckman and Ayers, 1994); it is useful to have an annotation object which corresponds directly to these types. When data are being extracted for these events, the system should return a single data point rather than the set of points returned for a segment.

4.1.1. Relational structure

Relationships can exist between tokens both within and between levels. The meaning of a relation between two tokens, and any constraints on the validity of such a relation depend upon the type of the tokens and should be largely defined by the application, rather than by the annotation system. A general purpose annotation system could be designed with only one kind of relation between tokens – the interpretation of which would be left entirely to the application. However, there are at least two kinds of relationships which are so frequently used in linguistic annotation that they merit special treatment: the *sequence* relation and the *domination* relation. The sequence relation, described earlier, relates tokens at the same level whereas the domination relation relates tokens to constituent tokens at the same or different levels. The system also allows a general purpose typed *association* relation which has no pre-defined interpretation and can be used to express application specific relations. These three relations are illustrated in Fig. 4.

In the Emu system, the meaning of the two kinds of inter-level relations is as follows:

- *Association relations* record an arbitrary association between tokens and may exist in any pattern within an utterance both within and between levels. An application may define any number of association types and the type name is part of the relation; for example, a *coreference* association might be used between words in a dialogue annotation. There are no constraints on, for example, the number of relations a token can take part in and multiple relations can be differentiated based on their types. Association relations are *directed* (if A is associated with B then B is not necessarily associated with A) and *intransitive* (A associates B and B associates C does not imply that A associates C).
- *Domination relations* exist between one token (the parent) and an *ordered set* of tokens

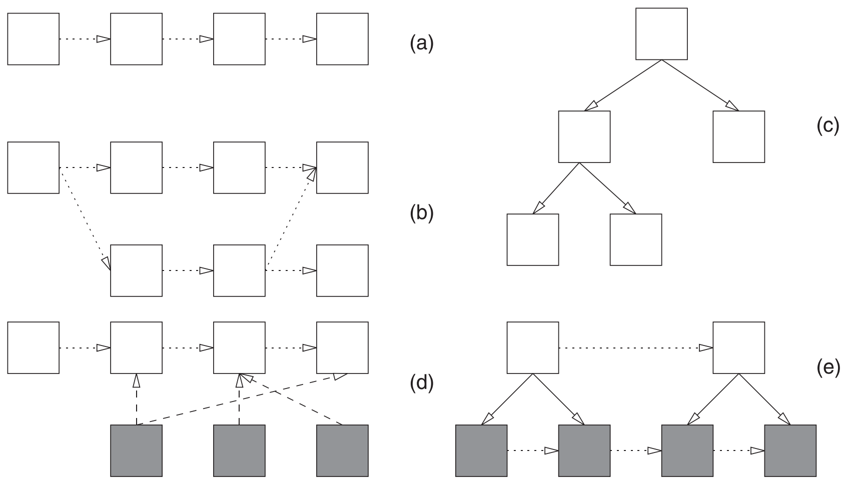

Fig. 4. The possible relations between tokens both within and between levels. (a) A simple sequence (dotted arrows). (b) A partial ordering as might be seen with overlapping speaker turns. (c) A recursive tree structure of domination relations (solid arrows). (d) An association relationship between different types (dashed arrows). (e) A domination relationship between different types.

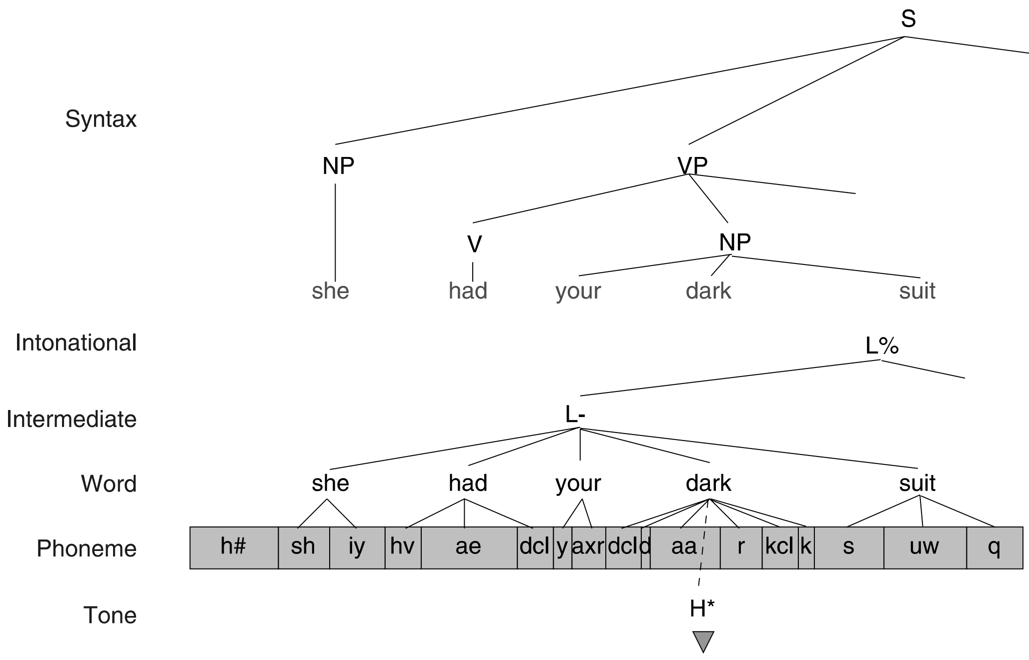

Fig. 5. An example of an Emu annotation. The Phoneme level contains segments with times from which the temporal information at other levels is derived. The Word level is dominated by two independent hierarchies and is duplicated here for clarity.

(the children). The relation implies temporal inclusion of the children within the parent and hence that the start and end times of the parent can be derived from those of the first and last children. Domination relations can exist both within and between levels; when they are within the same level they may form a recursive tree structure. No cycles are allowed in the domination graph. One parent may dominate different sets of children at different levels; for example, a syllable might dominate a set of phonemes as well as a set of pitch events. The domination relation is *directed* and *transitive* (A dominates B and B dominates C implies A dominates C).

The only restriction placed on the tokens taking part in a domination relation is that the addition of the relation must not introduce any ambiguity into the ordering relations within a level. This might arise if, for example, two different timed levels were linked to the same parent level in such a way as to provide ambiguous timing information.

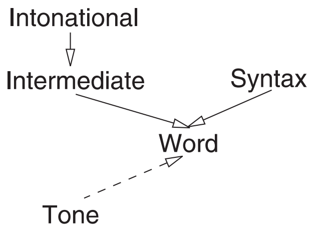

Fig. 6. The structure of the annotation of Fig. 5. Solid lines denote domination relations while dashed lines denote association relations. A line between levels indicates that these relations *may* exist between tokens at these levels.

An example of an Emu annotation is given in Fig. 5 with the corresponding annotation structure illustrated in Fig. 6. In this example, Tone and Phoneme levels have associated times and all other levels derive their times from the Phoneme level via domination relations. The Word level is dominated by two levels, one of which is part of the ToBI (Beckman and Ayers, 1994) style annotation and another which describes the phrase structure of the sentence.

4.2. The no-crossing constraint

There is a danger in presenting a generalised data model for linguistic annotation that the terms used to describe the model have well-defined meanings in certain fields of linguistics. For example, the use of *domination* and *association* to describe the relations permitted in the Emu system might imply to some users that Emu is based on a particular linguistic or phonological model where these terms have a theoretical meaning. As far as possible, Emu intends to be theory-neutral but to provide an environment that is rich enough to be used in a number of areas of linguistic research.

A case in point is the constraints placed on the *domination* relation in the Emu system. A common restriction on this kind of relation in some application areas is that the children of neighbouring parent tokens may only overlap at their edges: that is, only the leftmost and rightmost children may have more than one parent. This is known as the *no-crossing constraint* (Coleman and Local, 1991) as it implies that lines must not cross in the annotation. One might imagine then that Emu could enforce this constraint in order to allow only well-formed annotations.

However, enforcing this constraint would prevent annotating the kind of dialogue shown in Fig. 7. In this case, two sets of words have been encoded on the same level and then linked via domination relations to a speaker turn level. Clearly, domination lines cross in this situation, although the no-crossing constraint is not actually violated since each set of words would normally be considered a tier in itself. The solution is either to force the words from each speaker to be placed on a different level (Words1 and Words2) or to allow domination lines to cross in this way.

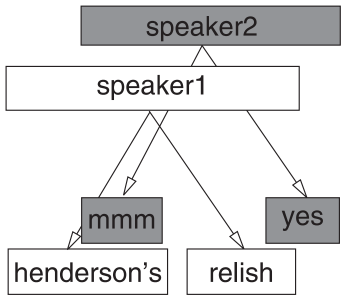

Fig. 7. An example of a two speaker dialog annotation showing overlap between speaker turns.

Emu attempts to provide an interpretation of the domination relation that will allow both these modes of use. By requiring only that domination relations introduce no ambiguity into the annotation, Emu allows lines to cross where the user deems that appropriate. By allowing scripts and other extensions, Emu can enable rules such as the no-crossing constraint to be enforced in particular applications.

4.3. The database template

The structure of every database managed by Emu must be specified in a *database template* file which describes the different levels of annotation and the allowed relationships between them. The template acts like the DTD (document type declaration) in SGML (Goldfarb, 1990) since it describes the shape of the annotation and the kinds of relations that are to be allowed between different kinds of tokens. The template file also describes other aspects of the database such as the location of the annotation files and configuration information for the Emu tools or user scripts.

Each level of annotation is listed in the template together with the other levels with which tokens at this level can be related. Any pair of levels may be related by either domination or association relations except that there must be no cycles of domination relations (i.e., if A dominates B then B cannot dominate A). Relations are also allowed within a level: for example, words may be associated with other words, or syntactic categories (such as noun phrase or verb phrase) could dominate each other.

By default, all annotations are stored in an Emu label file but parts of the annotation may also be stored in *external* label files in various formats. A common mode of use is to have the labels for individual levels stored in separate label files along with their timing information: for example, the TIMIT database provides phoneme and word level labels in this format. The Emu label file then stores references to these files and any additional annotation which may be overlaid on them. The template file contains details of the formats of these external files and the levels that they are associated with.

Emu uses a portable *utterance name* to identify the family of files within the database which belongs to one utterance. For each file type in the database (as defined by the file extension), the template lists a search path which can be used to locate these files. The search path can contain wildcards to match subdirectories. This facility enables databases to be moved to different computer systems retaining a common set of utterance names by changing only the database template. It is also possible to have parts of the database reside on different storage devices, for example, storing speech data on CDROM and label files on a hard disk.

5. Database queries

Central to the functionality of the Emu system is the ability to search and extract information from a corpus. Emu provides a rich annotation structure and the query system is designed to allow the specification of the desired tokens in terms of the token label and the position of the token in the relational structure of the annotation. The result of a query is a table of information about the tokens isolated by the query, with one row per matching token. In the current implementation, each row contains the token label, start and end times (or for events, a single time) and the utterance name that the token was found in; this information is sufficient to extract speech data corresponding to each token.

While this kind of return value serves the purpose of acoustic studies of particular kinds of segments well, it does not offer enough control in general; the kinds of results returned by database queries are currently under review (Cassidy and Bird, 2000) as discussed briefly later in this paper.

At the time of the design of the Emu query language, there were no publicly described systems which had a data model as rich as the Emu model. Those systems that did exist for multi-level annotation tended to use relational-like query languages (Kurematsu et al., 1990). The major inspiration for the Emu query language was the query system of MU+. In MU+, queries consisted of a set of conditions on tokens connected with *and* and *or* which constrained the label of the token or the label of one of its parents in the hierarchy. Thus the query

```
Phonetic = 'A'
  or (Phonetic = 'E' and Syllable = 'S')
```

selects Phonetic tokens labelled `A` or those labelled `E` that are dominated by syllables labelled `S`. Sequence constraints were specified using an index notation; for example, `Phonetic[-1]` referred to the phonetic token before the target token. The kind of tokens returned by the query was determined by a separate query option. The result of a query was the same table of labels and times as produced in the current system (Harrington et al., 1993).

While this query language served the kinds of acoustic phonetic studies carried out under MU+ its functionality was too limited. For example, it was not possible to constrain a sequence of two phoneme tokens to be dominated by the same word token. Our approach to the design of a new query language was to make reference to the explicit domination and sequence relations and to allow compound queries to be easily built from basic elements. After five years of use, it is now clear that this query language is not sufficiently expressive for the range of queries that might be put to a speech database and the query language is again under revision; this topic will be revisited later in this paper.

The query language described here is that implemented in the first version of Emu. Since that version lacked the association relation there is no way to query association in this language. Rather than introduce an ad-hoc extension to this language we prefer to present it in its present form and note that we are working on a new query language which will allow more flexibility (Cassidy and Bird, 2000).

5.1. Simple queries

Simple queries relate to a single token and constrain either the labels on the token or its

Table 1
Examples of simple and compound queries

Query	Comments
`Phonetic = p`	Find phonetic segments labelled p
`Phonetic = A|I|O|U|E|V`	A disjunction of labels
`Phonetic = vowel`	A label category defined in the template
`Start(Word, Syllable) = 1`	Matches any first syllable in a word
`Phoneme = vowel` & `Position(Word, Phoneme) < 2`	Vowels in the second position within a word
[`Phoneme = vowel` → `Phoneme = stop`]	Find a sequence of vowels followed by stop at the phoneme level
[`Syllable = S ^ Phoneme = vowel`]	Find syllables labelled S dominating vowel phonemes

position relative to its siblings. Queries can refer either directly to token labels or to the categories defined in the database template. Positional queries, such as `Start(Word, Phoneme) = 1`, succeed for children in the appropriate position relative to their siblings. The two levels in a positional query must be related by a domination relation. Different constraints can be conjoined with the & operator. Some example queries are shown in Table 1.

5.2. Compound queries

Simple queries can be combined to constrain the sequential, domination or association relations between tokens. The form of these queries is illustrated in the last three entries in Table 1. Since compound queries refer to more than one token, a decision must be made about which token the query as a whole returns; in the case of the sequential query, the start and end times of the sequence are returned with a label containing both token labels, whereas for the domination and association queries, the label and times of the first token in the query are returned. To modify this default action, one of the tokens can be marked with a hash (#) to indicate that it should be returned in the result table; for example, the query

```
[Syllable = S ^ #Phoneme = vowel]
```

returns a list of phoneme tokens and the query

```
[Syllable = S → #Tone = H*]
```

returns a list of Tone events.

Compound queries can be arbitrarily nested to specify complex constraints on tokens. As an example the following query finds sequences of stop and vowels dominated by strong syllables where the vowel dominates an `H*` tone target (note that domination is used here where association would be more appropriate since the earlier Emu system did not include the association relation): the result is a list of the vowel labels with associated start and end times.

```
[Syllable = S ^ [Phoneme = stop
    → [Phoneme = vowel ^ Tone = H*]]]
```

5.3. Query engine implementation

Since Emu has been used on relatively small databases, the current implementation of the query engine is able to use a linear search of the tokens in each utterance in the database in order to satisfy a query. While this is perhaps not the most efficient way to search a database, it results in acceptable query times even on moderately sized acoustic phonetic databases. In a recent study, comparing queries using the Emu search engine with those on an equivalently structured relational database (Cassidy, 1999), most searches using the Emu engine took around 40 s. The example database used in these experiments contained 1000 read sentences labelled at phonetic, phoneme, syllable, word and intermediate and intonational phrase levels, a total of 136,749 tokens and 438,559 links.

To execute a query, each utterance is read in turn and the query constraints are tested in all possible match positions in the annotation structure. Within an utterance, each of the simple parts of the query is matched against tokens at the appropriate level, so `Phoneme = vowel` would select all the vowel phoneme tokens. These lists are then combined using the constraints of the compound query elements; so if two simple queries are constrained to be in sequence, then only those pairs which are in sequence are retained. If at any time the list of candidate tokens reduces to zero, the match is terminated. At the end of processing the entire query, the targeted tokens are returned to the caller.

6. Discussion

The Emu speech database system is a set of software tools which facilitates the creation, management and use of annotated speech databases. Emu achieves this by providing a scripting environment tailored for manipulating annotations in a natural way, and a database query engine which can locate acoustic tokens based on sequential and relational constraints. Emu's design is intended to make the system useful for a wide range of speech databases irrespective of speech or label file formats.

6.1. Related work

This special issue of Speech Communication includes a number of general purpose speech annotation formalisms and systems. Three of these in particular (Bird and Liberman, 2001; Taylor et al., 2001; McKelvie et al., 2001) attempt to address similar goals to the Emu system; it is worthwhile reviewing them in relation to Emu.

Each of these papers presents a framework for linguistic annotation but only McKelvie et al. (2001) describe a system which is currently widely useable in speech corpus research. The Bird and Liberman proposal has widespread support and tools are now in active development. The formalism described by Taylor is part of the Festival speech synthesis system and while it offers very flexible representation mechanisms, it does not provide user level tools for creating, querying and accessing collections of annotated speech. The MATE project (McKelvie et al., 2001) is a fully developed corpus annotation system which focuses on the annotation of spoken dialogue and is used in a number of European Union projects. We believe that Emu is the only speech database system which provides for creation, management, querying and analysis of data from speech corpora that is widely available and in use in a range of speech database projects.

6.1.1. Annotation graphs

Annotation graphs (AGs) (Bird and Liberman, 1999, 2001) represent linguistic tokens as edges in a directed acyclic graph, where the nodes in the graph represent the boundaries between tokens which may be associated with a time reference. The equivalent of the Emu domination relation is represented by 'structural inclusion' of one node within another where, for example, a word arc will span the nodes corresponding to the boundaries of its constituent phonemes. This can be seen in Fig. 8 which shows the equivalent of Fig. 5 in the AG format. Labels on each arc contain two or three parts: a *type label* and a *token label*, separated by a forward slash and optionally an *equivalence class label* which can be used to represent association between arcs. This scheme has the advantage of being able to represent *n*-ary relationships simply, such as those between sets of words in a translated text and the original.

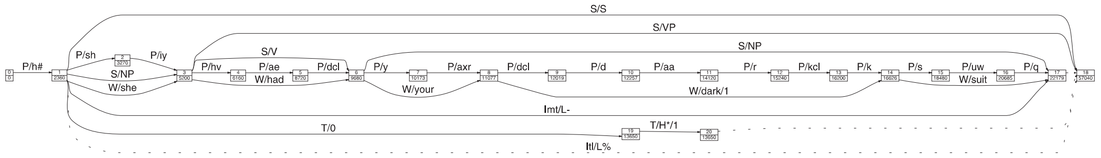

Fig. 8. An example of an Annotation Graph (Bird and Liberman, 1999) corresponding to the augmented TIMIT example from Fig. 5.

The authors propose a set of indices on annotation graphs which would facilitate searching for particular kinds of object or relation. Hence an equivalence class index would make finding inter-token relations efficient and an index built on implicit hierarchical structure could help in answering domination style queries. Their argument is that the annotation graph framework allows the representation of any linguistic structure at least implicitly, and that indices can be used to make relevant aspects of that structure efficiently accessible.

As with the Emu system, Bird and Liberman propose a set of input filters that will convert an existing database annotation into the annotation graph format for internal manipulation. Work is in progress on a query language which will provide search facilities on annotation graphs (Cassidy and Bird, 2000; Bird et al., 2000); importantly, the results of a query will be a well formed annotation graph, allowing subsetting of databases in a useful way.

Bird and Liberman provide an extensive review of linguistic annotation in their paper as well as compelling arguments that the annotation graph formalism can represent a wide range of annotation types used in speech and language corpora. Although their structures might seem radically different to those used by the Emu system, in fact, they are almost entirely equivalent. Fig. 9 presents a sketch of a procedure for converting AG annotations into those for the Emu system with no loss of information. Each token within an Emu level maps to a labelled arc with corresponding start

```
To convert an Annotation Graph into an Emu annotation:

for each AG type
    create an Emu level for the type
    for each arc sequence of this type
        for each arc in the sequence
            append an Emu token with start and end times (if present)
        end
    end
end
create domination relations between structurally included tokens
create association relations from equivalence classes
```

Fig. 9. Pseudocode for converting an Annotation Graph (Bird and Liberman, 2001) annotation into one for the Emu system.

and end nodes; the times on the nodes are optional in each formalism. Both systems define sequence in the same way. In each case, a sequence relation defined by the user is not allowed to contradict the temporal ordering of tokens. Domination relations are implicit in annotation graphs being defined by structural inclusion; an index is proposed to provide efficient access to, for example, the children of a given token. Arbitrary relations between tokens are represented in the AG formalism via equivalence class labels: two or more arcs can be given the same label to denote a relationship between them. While Emu is only able to represent binary association relations, an application could equally well use the shared label mechanism to implement *n*-ary relations if needed.

The ATLAS project [http://www.itl.nist.gov/iaui/894.01/atlas/] has recently developed a C++ library which implements the annotation graph storage model. Since the AG model provides equivalent constructs for most if not all parts of the Emu storage model, we are now working on implementing Emu on top of this library. This approach allows us to take advantage of the work being done by various groups on the ATLAS project and maintain compatability with these tools. The Emu interface will initially be very similar to the current system and we hope to be able to provide input to the ATLAS project from our experience in using this library.

6.1.2. Heterogeneous relation graphs

As part of the Festival speech synthesis system, Taylor and others (Taylor et al., 2001) have developed a framework for representing annotations of speech signals which is used for communication between modules in the synthesiser. This framework, dubbed heterogeneous relation graphs (HRG), is described in this special issue as another general purpose annotation framework for speech database applications. Although HRGs are used within the Festival system we are not aware of any corpus creation or analysis tools that make use of this model.

An HRG consists of a set of *linguistic items* (corresponding to Emu tokens) which exist in *relation structures*; essentially an HRG is a graph connecting linguistic items by arbitrary binary relations. Each item can take part in many relations and the type of an item is not defined explicitly but is instead implicit in the relations that the item takes part in. Items themselves are implemented as attribute value lists (AVLs) and so can contain arbitrarily complex descriptions of linguistic objects.

The values within the AVL that represent a linguistic item can be either simple strings or numbers or they can be AVLs themselves. This recursive structure makes the HRG model very flexible as befits its use as an internal representation model within a speech synthesis system. These values can also be computed when needed by functions written in Scheme (a dialect of the LISP programming language); pre-written functions are provided to compute start and end times of items in various ways, e.g., from the next item in a sequence or from child items in a hierarchical relation. Again this mechanism is very powerful and allows the HRG mechanism to be adapted to many new kinds of application when the appropriate functions are in place.

In comparison to the Emu model, HRGs take a different approach to the problem of representing linguistic annotations. As befits the primary use of HRGs, they are well suited to storing arbitrary pieces of data which are actively updated and processed by the elements of a speech technology system. It is clear that the model has all the power of the recursive AVL data structures which will facilitate the storage of almost arbitrary data structures: hence it is unlikely that there will be linguistic annotation structures that cannot be stored in an HRG. The Emu model by comparison is more rigid in that annotation structure is predefined by the database template and specific mechanisms are provided to facilitate the representation of linguistic annotations. Emu provides one model of the domination relation which we believe captures the use of this construct in general linguistic annotation. The HRG model can implement a version of this relation, and provides library functions which make this easy to do, but domination has no more status in the formalism than any other kind of relation.

While the HRG model provides a very powerful formalism for representing linguistic annotation, much of the flexibility is useful primarily in the context of a speech technology system rather than the more static annotations of a speech database. It remains to be seen whether the additional functionality of HRGs beyond those in Emu and Annotation Graphs will provide useful tools for creation and analysis of speech and language corpora.

6.1.3. MATE

The MATE project [http://mate.nis.sdu.dk] (McKelvie et al., 2001) provides a set of tools built around XML and related standards for document markup. MATE uses an internal annotation format that is capable of representing arbitrary directed graphs. The primary focus of MATE is on the annotation of spoken dialog; the user level tools provided are oriented to these kinds of application. The MATE project also provides annotation guidelines designed to facilitate the exchange of data between researchers in Spoken Language Dialog Systems.

The data model provided by the MATE project appears to offer representational power equal to that of the Emu and Annotation Graph data models. MATE is closely tied to the XML standard which is used for all external data storage. Since XML provides a purely hierarchical data model, MATE uses the technique of standoff markup (Isard et al., 1998) which stores separate hierarchies in different files and uses XML pointers to record relations between the hierarchies. This use of a tool-specific data format is in contrast to the Emu approach which, while it does provide a native file format, aims to read many different file formats and provide a mechanism for including new file format handlers.

Although there are a number of papers describing the MATE system, it is unlikely to become a standard tool for corpus based research (beyond its use in European Union projects) until it becomes publicly available either as a commercial system or as free software.

6.2. Query languages

Neither Bird and Liberman (1999) nor Taylor et al. (2001) describe a query system for their frameworks, although a proposal for an Annotation Graph query language is being developed (Cassidy and Bird, 2000; Bird et al., 2000). MATE (McKelvie et al., 2001) includes a very well-developed query language, Q4M, which supports relational style queries (find all X, Y, Z such that X overlaps Y etc.). Query results are presented as XML documents and so are complete annotations in themselves which are linked to the original corpus via XML pointers. This is a very powerful mechanism which makes search results maximally useful: search results may be treated as a database in themselves retaining all contextual information from the original corpus.

The Emu query language was derived from that used by the MU+ system (Harrington et al., 1993) and serves the basic purpose of selecting tokens from the database satisfying various constraints on their labels and sequential and relational positions. The return value of a query is a list of matching tokens. For most simple acoustic phonetic applications this query language is adequate, however, it is becoming apparent that new applications of Emu will need a richer query language. Two kinds of extension can be identified: more flexible constraints and a mechanism for specifying the kind of result returned from a query.

The basic building blocks of the current query language enable queries to refer to single tokens or groups of tokens combined with sequence or domination/association relations. There is no mechanism for specifying optional tokens or sequences of one or more tokens in a query: for example, there is a way to express "find all syllables between a strong syllable associated with an H* tone and the end of the intermediate phrase". The language also needs to be able to combine queries with conjunction or disjunction: "find all vowels either preceded by a stop or at the start of a word".

It would be useful for a query system to return a result that is a valid annotation in itself: a subset of the original annotation of an utterance containing just those elements that matched the query. In this model, a report generator could then scan this subset annotation to produce any kind of output for the analysis stage.

The details of these proposed extensions to the Emu query language are still to be defined. There is a great deal of interest now in the database literature on query languages for semi-structured data (Buneman et al., 1998; Liefke, 1999; Deutsch et al., 1998) and in the document markup community in relation to XML (Bray et al., 1998; Clark and DeRose, 1999) all of which are relevant to the design of a new query language. We are actively researching this area and hope to make some proposals in the near future.

6.3. New applications

Having argued the generality of the annotation structures within Emu, it is natural to look at extending the range of application areas of the system within corpus based linguistic research. An exciting area of recent development is the annotation of dialogue both at a high level involving dialogue acts and speaker turns (Core and Allen, 1997) and at a level of intonational structure. Dialogue annotation is not usually time-aligned; instead a transcription is performed with many diacritical markers for events such as speaker overlap or word stress (Core and Allen, 1997); some recent standards have used SGML/XML style annotation and include time alignment at various reference points through the dialogue. Time alignment to audio and video recordings has also recently been introduced into the CHILDES (MacWhinney, 1995) project.

Annotation of long dialogues requires a different set of tools to those provided by Emu for annotation of short utterances. Although Emu is capable of representing the annotations internally, the Emu labeller was designed for detailed analysis of short segments of speech and so a new kind of user interface is required. A number of projects (MacWhinney, 1995; Flammia and Zue, 1995; Barras et al., 1998) have developed tools which support the transcription of long recordings, including time alignment at major event boundaries such as speaker turns. Annotations are saved in a variety of formats. These tools do not support the more detailed annotation required for detailed acoustic analysis. Instead of providing equivalent tools within the Emu framework, a more useful approach might be to take the high level annotation provided by one of these tools and use it to segment a long recording into many smaller utterances. These utterances could then be annotated using the existing Emu labeller; Emu could then merge the high level and low level annotations. This model again retains maximum backward compatibility with other tools and corpus formats in the spirit of standoff markup (Thompson and McKelvie, 1997).

A number of other planned developments of the Emu system can be mentioned here. Support for reading annotations using Unicode international character encodings [http:/www.unicode.org/] will allow natural annotation of non-English language materials. This extension should include the ability to display Unicode characters in the labeller and hence might also be used to display IPA characters instead of the more usual machine readable phonetic alphabet even if the underlying representation uses the latter. Implementation of this facility is aided by the recent inclusion of Unicode support in the Tcl/Tk toolkit (Ousterhout, 1994).

With the now ubiquitous access to the Internet, many groups are considering providing access to language corpora via the World Wide Web or another network interface. Such a facility would be very valuable for teaching and research as an alternative to the unrestricted distribution of large corpora which might be undesirable or simply impractical. Emu can potentially act as a remote server which answers database queries and returns the results to a remote client for analysis. We have experimented with simple Tcl-based client programs which query and extract data from a remote database and display the results either as a standalone application or embedded in a web page.

7. Conclusion

In combination with the Splus statistical package [http://www.mathsoft.com/splus], Emu provides an environment which supports detailed examination of numerical data associated with linguistic phenomena. Emu is freely available in source form from our web site [http://www.shlrc.mq.edu.au/emu/] and runs on Microsoft Windows and various Unix platforms; a port to the Macintosh system is in progress.

This paper has described the internal representation of linguistic annotations in the Emu system, arguing that they are sufficiently rich to allow Emu to deal with most existing annotated corpora. Further development is needed in the core system to deal with the different file formats and user interface requirements of these corpora.

Acknowledgements

We would like to thank an anonymous reviewer for comments on an earlier draft of this paper. The development of Emu would not have been possible without the support of various projects within the Speech Hearing and Language Research Centre and the feedback provided by Emu users around the world.

References

Altosaar, T., Millar, B., Vainio, M., 1999. Relational vs. Object-oriented models for representing speech: a comparison using ANDOSL data. In: Proceedings of the Eurospeech 99. Budapest, Hungary, pp. 915–918.

Barras, C., Geoffrois, E., Wu, Z., Liberman, M., 1998. Transcriber: a free tool for segmenting, labeling and transcribing speech. In: Proceedings of the First International Conference on Language Resources and Evaluation (LREC). Granada, Spain, pp. 1373–1376.

Beckman, M.E., Ayers, G.M., 1994. Guidelines for ToBI labeling version 2.0, [tobi@ling.ohio-state.edu] and [http://ling.ohio-state.edu/Phonetics].

Bird, S., Liberman, M., 1999. Towards a formal framework for linguistic annotation. Technical Report No. MS-CIS-99-01. Philadelpha, PA: Department of Computer and Information Science, University of Pennsylvania.

Bird, S., Liberman, M., 2001. A formal framework for linguistics annotation, Speech Communication 33 (1–2) 23–60.

Bird, S., Buneman, P., Tan, W.C., 2000. Towards a query language for annotation graphs. In: Proceedings of the LREC 2000, Athens, Greece.

Bray, T., Paoli, J., Sperberg-McQueen, C.M., 1998. Extensible markup language (XML) 1.0. available at [http://www.w3.org/TR/REC-xml] (W3C Recommendation), February.

Buneman, P., Deutsch, A., Fan, W., Liefke, H., Sahuguet, A., Tan, W.C., 1998. Beyond XML query languages. In: Query Language Workshop (QL'98).

Carlson, R., Granström, B., Nord, L., 1990. The KTH speech database. Speech Communication 9, 375–380.

Cassidy, S., 1999. Compiling multi-tiered speech databases into the relational model: experiments with the Emu system. In: Proceedings of Eurospeech 99. Budapest, Hungary, pp. 2239–2242.

Cassidy, S., Bird, S., 2000. Querying databases of annotated speech, In: Orlowska, M.E. (Ed.), Proceedings of 11th Australasian Database Conference. Canberra, Australia, Vol. 22, pp. 12–20.

Cassidy, S., Harrington, J., 1996. EMU: an enhanced hierarchical speech data management system. In: Proceedings of the 6th International Conference on Speech Science and Technology. Adelaide, pp. 361–366.

Clark, J., DeRose, S., 1999. XML path language (XPath) version 1.0. available at [http://www.w3.org/TR/xpath] (W3C Working Draft), July.

Coleman, J.S., Local, J.K., 1991. The no crossing constraint. Autosegmental Phonology Linguistics and Philosophy 14, 295–338.

Core, M., Allen, J., 1997. Coding dialogs with the DAMSL annotation scheme. In: AAAI Fall Symposium on Communicative Action in Humans and Machines, Boston, MA.

Crawford, M., Brown, G.J., Cooke, M., Green, P., 1994. Design, collection and analysis of a multi-simultaneous-speaker corpus. In: Proceedings of the Institute of Acoustics. Vol. 16, pp. 183–190.

Deutsch, A., Fernandez, M., Florescu, D., Levy, A., Suciu, D., 1998. XML-QL: a query language for XML. Available as [http://www.w3.org/TR/NOTE-xml-ql] (W3C Note), August.

Flammia, G., Zue, V., 1995. N.b.: A graphical user interface for annotating spoken dialogue. In: Moore, J., Walker, M. (Eds.), Empirical methods in discourse interpretation and generation. Papers from the 1995 AAAI Symposium. pp. 40–46. Available at [http://sls-ww.lcs.mit.edu/flammia/Nb.html].

Goldfarb, C.F., 1990. The SGML Handbook. Clarendon Press, Oxford.

Harrington, J., Cassidy, S., Fletcher, J., McVeigh, A., 1993. The MU+ system for corpus based speech research. Computer Speech and Language 7, 305–331.

Hedelin, P., Huber, D., 1990. The CTH speech database: An integrated multilevel approach. Speech Communication 9, 365–373.

Hendriks, J.P.M., 1990. A formalism for speech database access. Speech Communication 9, 381–388.

Hieronymus, J.L. 1994. ASCII phonetic symbols for the world's languages: Worldbet. Technical Memo. AT&T Bell Laboratories.

Isard, A., McKelvie, D., Thompson, H.S. 1998. Towards a minimal standard for dialogue transcripts: a new SGML architecture for the HCRC map task corpus. In: Proceedings of the Fifth International Conference on Spoken Language Processing. Sydney, Australia.

Kurematsu, A., Tekeda, K., Sagisaka, Y., Katagiri, S., Kuwabara, H., Shikano, K., 1990. ATR Japanese speech database as a tool of speech recognition and synthesis. Speech Communication 9, 357–363.

Liefke, H., 1999. Horizontal query optimization on ordered semistructured data. In: WebDB'99.

MacWhinney, B., 1995. The CHILDES Project: Tools for Analyzing Talk, Second Edn. Lawrence Erlbaum, Mahwah, NJ.

McKelvie, D., Isard, A., Mengel, A., Møller, M.B., Grosse, M., Klein, M., 2001. The MATE workbench – An annotation tool for XML coded speech corpora. Speech Communication 33 (1–2) 97–112.

Ousterhout, J.K., 1994. Tcl and the Tk Toolkit. Addison Wesley, Wokingham, UK.

Schiel, F., Burger, S., Geumann, A., Weilhammer, K., 1998. The Partitur format at BAS. In: Proceedings of the First International Conference on Language Resources and Evaluation. Granada, Spain.

Taylor, P., Black, A.W., Caley, R., King, S. 1998. Edinburgh Speech Tools Library. [http://www.cstr.ed.ac.uk/projects/speechtools.html].

Taylor, P., Black, A.W., Caley, R., 2001. Heterogeneous relation graphs as a formalism for representing linguistic information. Speech Communication 33 (1–2) 153–174.

Thompson, H.S., McKelvie, D., 1997. Hyperlink semantics for standoff markup of read-only documents. In: SGML Europe '97. [http://www.ltg.ed.ac.uk/ ht/sgmleu97.html].

Watson, C., Harrington, J., 1999. Acoustic evidence for dynamic formant trajectories in Australian English vowels. Journal of the Acoustical Society of America 106, 458–468.