The MATE Annotation Workbench: User Requirements

Jean Carletta and Amy Isard
HCRC, University of Edinburgh
J.Carletta@ed.ac.uk, Amy.Isard@ed.ac.uk

Abstract

The MATE project (Telematics LE4-8370) aims to facilitate the re-use of spoken dialogue resources and to foster empirical investigation of dialogue by providing a workbench which can be used to annotate and explore the relationships among different structures within a dialogue corpus. This paper describes the intended functionality of the workbench by reference to the needs of several types of prospective users. It should be considered a position paper about what kind of technological support the user community requires. The workbench itself is scheduled to be released in December 1999, with further development likely beyond that.

1 Introduction

Many people wish to annotate spoken dialogue corpora with coded information. This information can come in many forms for many different purposes. Some of it will be linguistic; for instance, the annotation may represent part-of-speech information or syntactic structure, for use in language modelling. Some of it will be non-linguistic, representing information about the communicative situation or about events such as coughing or gesturing. If the dialogue being annotated is being conducted with or mediated by technology, some of it may be specific to that technology – for instance, showing the results of speech recognition in line with a human transcription of the same material in order to highlight where the dialogue model broke down. Although some kinds of annotation, such as dialogue act information, come up again and again, it is impossible to prejudge what kinds of annotations people who work with corpora will find useful even when considering a quite restricted set of coding purposes.

Currently, corpus annotation is a very costly exercise not just because of the time which it takes coders to make the coding distinctions, but also because there is little effective technological support for annotation. The MATE Workbench [9] is intended to address the need for technology by providing a single interface to all of the basic functionalities which corpus annotators need, but with enough flexibility that different projects can to provide different kinds of annotation and that information can be ported between the Workbench and other applications. The workbench is a standalone tool written in Java, so it will be able to run without recompilation on many different platforms including PCs, Macs, and Unix machines.

2 Why This is Hard

The single most important obstacle to annotation tool design is the fact that corpus annotations are not necessarily hierarchically arranged, making it difficult to design data structures and algorithms which can handle them efficiently but which are also flexible enough to make it easy to implement new annotation schemes. For instance, dialogue might usefully be annotated for intentional structure, intonational structure, *and* syntactic structure, and, in fact, for the most detailed empirical work, all of these annotations might be needed simultaneously so that relationships among them can be explored. Each type of structure can refer to some shared base level of transcription, whether orthographic or phonetic, with timing information where the speech signal is available. Each type of structure might involve several kinds of tags (for instance, dialogue moves and games, or part-of-speech labelled tokens, phrases, and sentences) but can be considered to be broadly hierarchical. However, the hierarchies themselves may or may not bear any relationship to each other, as in Figure 1.

Although it is possible to design tools which can handle such complex arrangements of tags, it is much easier to do so when a concrete set of tags is envisioned than to implement a generic solution which can be adapted to other tag sets. Therefore, existing tools, (reviewed in MATE deliverable 3.1 [2]), such as DAT [8] tend to support particular coding schemes, at best allowing tags to be renamed or extra categories to be added to a tag set within a strict family resemblance (for example, nB [5]), and, in fact, few implement sets of tags which cross in this way. In addition, existing tools tend to have fixed methods for displaying information and fixed

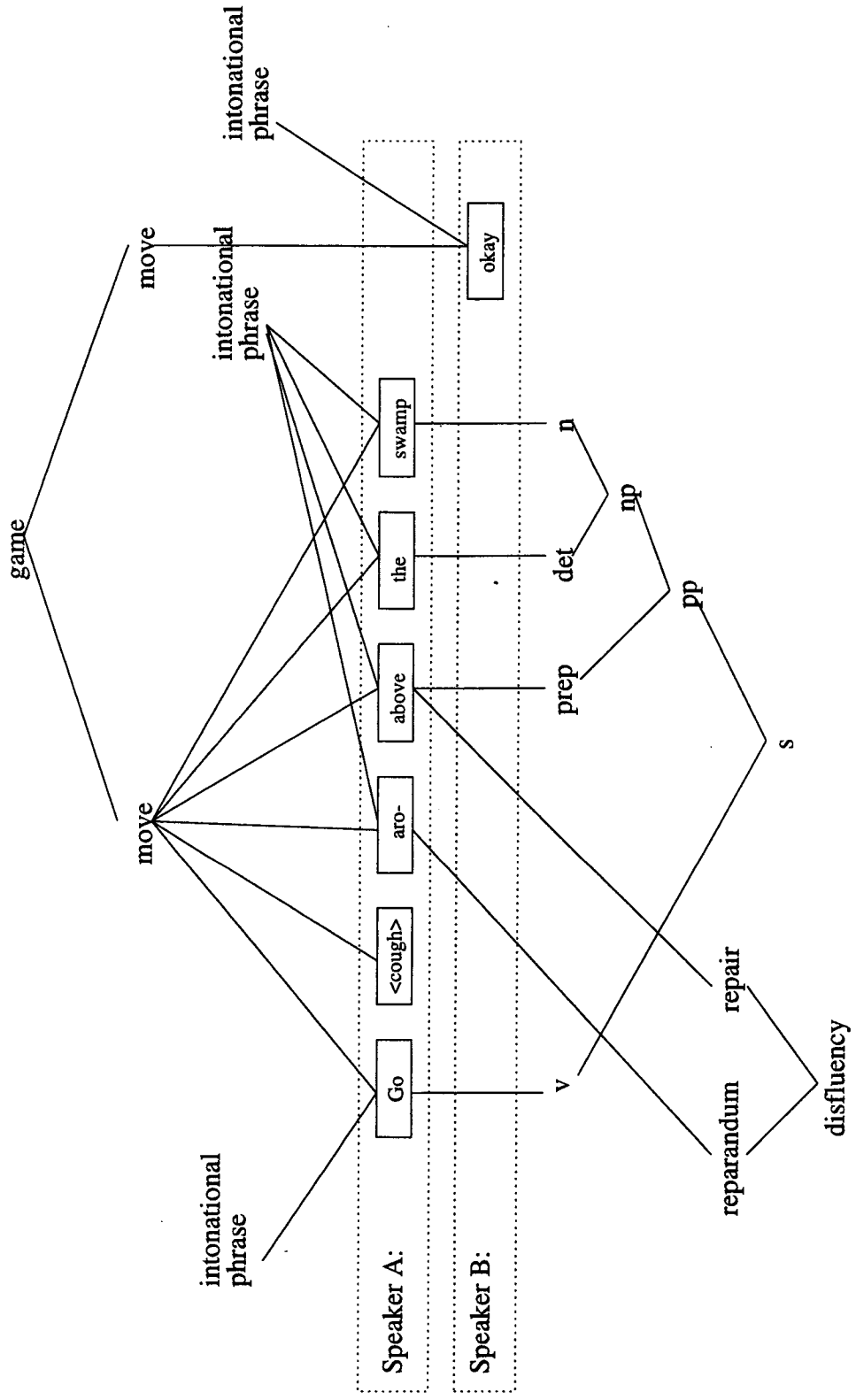

Figure 1: Overlapping Hierarchies

actions for coding. Since tools are often built for particular coding exercises, the user interface is also fixed, with no way of altering it to suit personal preferences. Although these limitations are understandable, they are not insurmountable. The MATE Workbench aims to improve on the current state-of-the-art by exploiting recent developments in corpus representation which allow a more flexible solution.

3 A Step Forward, but not Magic

The technological solution to the flexible representation of overlapping tag hierarchies is to use the XML markup language [11]. XML can be used to describe any sort of coding, and the coding structure can be described in a Document Type Definition (DTD) which describes what tags are possible, and where they can occur. It is not possible for XML tags to overlap within a document, but one can create overlapping hierarchies of markup in the same corpus by using "hyperlinks", which serve not just to point to elements in a different hierarchy of markup but also to include them for structural purposes [10]. For example, if dialogue moves are made up of words, and sentences are made up of the same words, as in Figure 1, then both can be linked to the same copy of the words, providing a way of relating the two structures. An example XML representation of a short dialogue fragment is given in Figure 2.

Of course, XML is designed to be machine-readable, and should rarely (if ever) be inspected directly. The MATE project will use "stylesheets" based on the XSL language [12], which allow one to express operations on an XML corpus which can be used to typeset it for the human reader, choosing both what annotations to make visible and how they should be displayed. Using MATE stylesheets, it will also be possible to specify the link between user actions and modifications to the XML which is being displayed, and thus implement coding interfaces. Stylesheets are written as a sequence of rules which match the input and give instructions about what to do for the output. A simple example of such a rule is:

```
<msl:template match="($a: dummy);"> (1)
   <result>                          (2)
      <msl:text>A dummy.</msl:text>  (3)
   </result>
</msl:template>
```

Line (1) introduces one stylesheet rule. There will usually be several rules in a stylesheet. The 'match' attribute tells us which elements this rule will apply to. In this case, the rule will match `<dummy>` elements in the input document. In the MATE stylesheet language, matching is done using the MATE query language; query construction is supported in the workbench by means of a graphical user interface. Line (2) and following lines gives the template of the rule. This is a description of the elements which will be created in the output document. In this case `<result>` is a literal element, and the result of this rule will be that all `<dummy>` elements will be converted into

```
<result>A dummy.</result>
```

elements.

Rules in a stylesheet are applied in order of occurrence starting with the top level element in the document. There is a mechanism for top-down left-to-right traversal of the input document hierarchy, and a default rule for unmatched elements.

Using XML and XSL allows a flexible solution to the problem of technological support, but this solution is not magical. For instance, XML and XSL do not remove the need to understand how tags relate to each other; they simply make it easier to specify a good machine-readable representation of complex tag relationships and to display these relationships for the human reader. To see what benefits this technology will bring, it is necessary to analyse the capabilities and requirements of different types of potential workbench users separately.

4 User Types

Type 1: The Coder

At least for sites involved in large scale coding exercises, data coders are typically the cheapest labour source available. They do not wish to know anything about how the coding interface works or even how different sets of tags relate to each other. Their needs are fairly simple: an intuitive coding interface so that they can concentrate on the code distinctions, documentation of how to use the interface – although, human nature being what it is, we find that no amount of written material will replace good verbal instruction – and the coding instructions nearby, preferably on-line. The MATE Workbench will encourage fulfilment of these needs by providing guidelines and slots for documentation within the coding modules which define the coding task, through the example coding schemes which come with with the workbench, and by providing sufficient coding actions for a wide range of interface designs. Thus the main benefit to coders is simply a side effect of having good support for the implementation of coding schemes via well-defined coding modules. Another potential benefit for longer-term coders is the ability to reconfigure the user interface which controls interaction with the workbench components to suit personal preferences, for instance, by rearranging the menus and buttons.

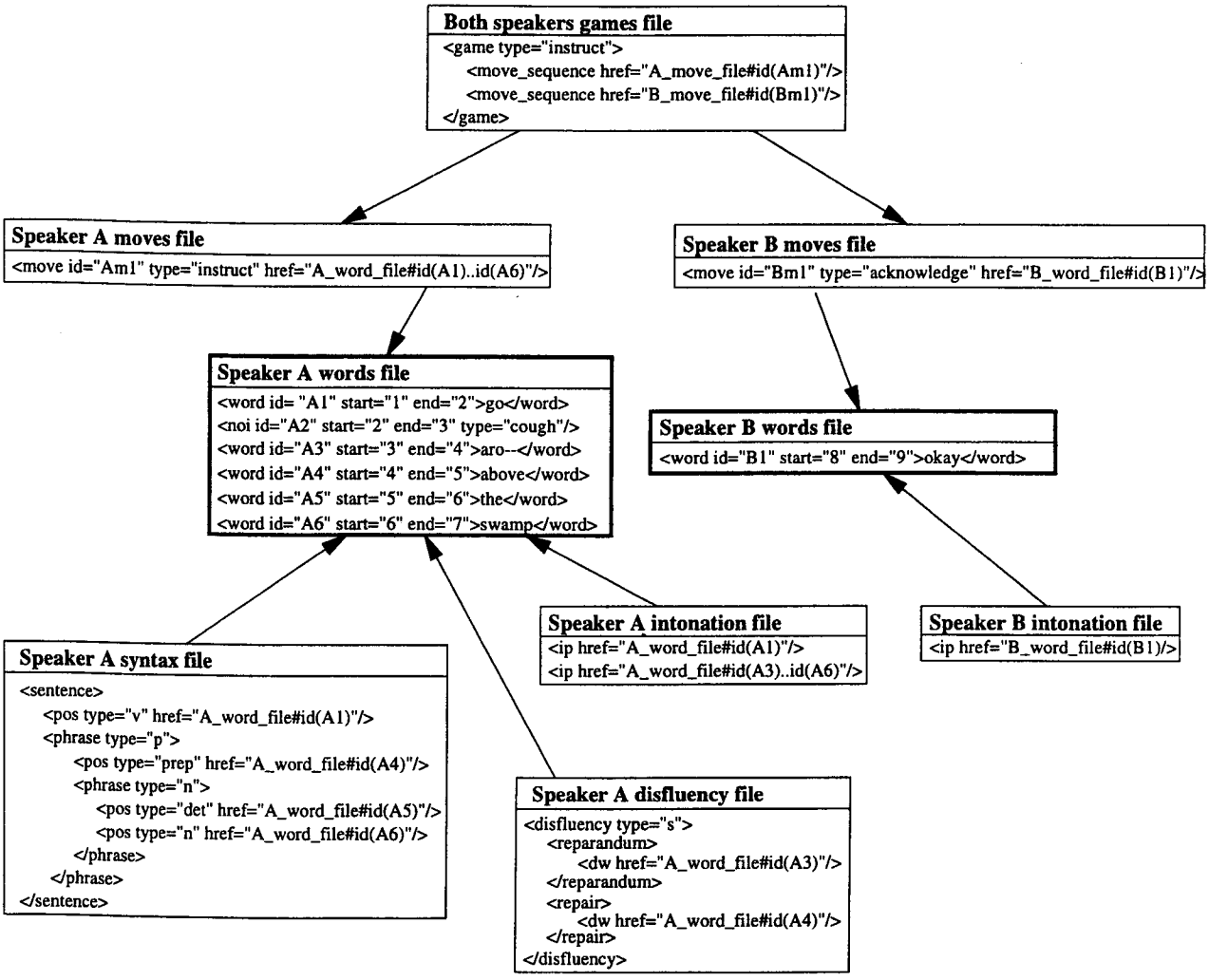

Figure 2: XML File Structure

Type 2: The Coding Consumer

There are several possible types of consumers of existing coded data. Some users might wish to check the relationships in the data for things which are statistically aberrant, mining the data pre-theoretically for whatever stands out. Others might wish to export some part of the data in order to train on the relationships which are present in it, in the hopes that some theoretically-motivated relationship will improve performance in, for instance, a spoken dialogue system. Straight theoreticians might wish to inspect particular relationships in order to test specific research hypotheses. Whatever the reason for interest in the corpus, consumers are united in their need to ask questions of the corpus, looking for places which match a specific form, and to display the results. Using the coded data well requires the mathematical capacity to understand the kind of structural information represented graphically in Figure 1, since otherwise the questions which the user asks will be meaningless. The more theoretically motivated the user, the more important this requirement is. Of course, this is true for all work with complex tag sets, not just those represented within MATE. In addition, where new kinds of display are required to match specific explorations of the data, new stylesheets will be required. The main benefits of the workbench for coding consumers are (a) the possibility of combining many different kinds of annotation on one data source, (b) a well-specified query language for exploring the relationships among the tags, and (c) methods for exporting different cuts on the data to other packages for further theoretical or statistical analysis.

Type 3: The Coding Developer

Many people wish to design their own coding schemes, either to improve on the reliability or suitability of an existing scheme or in order to test a particular research question. These coding developers may hire type 1 coders, but they are quite likely to do their coding themselves. This group of users has the hardest job. Designing a complete corpus requires the mathematical capacity not just for understanding structures such as that represented in Figure 1, but also for constructing new ones and mapping these into the sorts of file structures represented in Figure 2. This is true whether or not the corpus is represented in XML but is sometimes hidden away as something which only the software developer truly understands. One need not understand the relationships among all the tags on a corpus in order to install a new coding level, but one must at least be able to hook a new tag set into some part of the existing structure. Although this may seem onerous for the user, in reality most of the requirements are the same as they were before; users who wish to do something new have to understand what it is they are trying to do. The only additional requirement is that instead of developing their own ad hoc data representations and mappings between the data, the screen, and user actions, users need to understand how data is represented in XML and how to write stylesheets expressing these mappings. The benefit of XML and XSL for this activity is that they are likely to be better-structured and more flexible than anything coding designers will come up with for themselves, especially if they are not experienced software developers, and that there are many existing tools for support, with more on the way.

5 Our Solution: Pre-Implemented Schemes Plus Development Tools

It is difficult to address all the user types at once, and since so little support currently exists for coding designers, it will be impossible to get the facilities for them right the first time. The fact that there are many users who do not necessarily want to implement their own coding schemes leads us to a staged solution. The MATE Workbench will be distributed with coding tools and basic display capabilities for a range of coding schemes at various levels of annotation from prosody and morphosyntax to dialogue acts and communication problems. These schemes are being chosen (or, in some cases, developed), based on an extensive review [4]. In addition to simply being practical and reliable schemes to represent their levels of coding, they are being chosen to represent as wide a spread of coding types as possible in order to test the workbench design. Current coding tools have also been reviewed [2] in order to inform us both about good design for tools for the chosen schemes and about the range of capabilities needed in the workbench. These schemes and the tools implemented for them can be used to allow users to develop a sense of the workbench capabilities, and for users who do not wish to implement new coding modules, they may be all that is required. The introduction of new schemes will be supported by tools for authoring XML corpora and stylesheets, and by the use of the existing scheme implementations as examples to modify.

6 Basic Functionalities

Keeping in mind our description of the basic user types, these are the basic functions which the MATE workbench will provide.

6.1 Display

Given a coded corpus, the display function will show on screen a human-readable version of the data. This display will be produced from the data using an XSL stylesheet, although there is no reason for the user to know that. Display options will include the size and placement of windows, text colour, font, and size, and text layout such as lists and tables. The display may include a speech waveform if one is associated with the dialogue, and user actions (such as clicking on an area of text) may be associated with further display information.

Some users believe that our flexible approach to document typesetting means that conceptualising complex relationships within a data set will become easier – that is, that using the workbench will help to clarify their thinking about what to look for. In a sense, it will, because the ability to write specialist display stylesheets will allow the user to create views which abstract away to the right sections of the data. However, stylesheets do not enforce the creation of a usable data display. In particular, it is just as possible to overload a display with too much information using this technique as using any other (and perhaps more tempting, because the stylesheets make this easier to do). The basic limitation on conceptualising data relationships is human, and not a product of coding technologies.

6.2 Query

Given a coded corpus, the query function will allow the user to construct a query which will match some part of the data set, and then will either extract that part (which can then either be exported or sent via a stylesheet for display) or count the number of matches, for performing frequency analyses. The MATE query language [6, 7, 3] contains constructs which allow the expression of either hierarchical or temporal relationships among a set of tags. This means that structural constraints can be given naturally (such as asking for all response moves within

dialogues of a particular type, or verbs within relative clauses), but that cross-hierarchy constraints can also be expressed (such as asking for all disfluencies which occur during a particular type of intonational phrase). The MATE workbench includes a point-and-click query formulation support tool; alternatively, queries can be typed at a command line.

6.3 Coding

Coding tools will allow the user to add an annotation corresponding to a particular coding scheme. Coding interfaces will be specified by means of stylesheets. Typical coding actions might include using the mouse to specify a location, to sweep areas of text, or to bring up a text window or menu by which tagging details can be entered.

6.4 Transcription

The transcription process should be highly individualised for a particular corpus depending on how recordings were obtained and for what purposes the corpus has been collected. Getting the process wrong can add months and great expense to a project. Good transcription requires software, such as spelling checkers, which would be difficult to provide within a Workbench. In addition, even if good transcription tools were supported by the workbench, many projects would not be able to use them because they contract out transcription work to secretarial agencies which are only willing to quote for the work based on the model of audio-typing using standard word processing packages. As a result, we would expect most users to wish to do their transcription elsewhere, using other software, and to transfer their transcripts into the workbench when they are complete. On the other hand, many users experimenting the system will wish to input small amounts of data so that they can test out the coding schemes and the workbench on new materials. In the first instance, we intend to provide a very simple transcription facility which will suffice for this purpose but which one would not wish to use for large-scale transcription, at least not without a great deal of thought about the alternatives.

6.5 Import

The more existing corpora the workbench will work with, the more useful it will be when it is introduced. Unfortunately, current corpora are in a wide range of formats, many of which bear little relationship to XML. We intend to supply two conversion tools with the workbench which will handle conversion from BAS-Partitur and Entropic xwaves xlabel files into XML. Corpora produced to EAGLES recommendations [1] require minimal conversions. Users of other formats must support their own conversion processes, the software for which can be installed in local copies of the workbench.

6.6 Export

Just as users of existing corpora may wish to import data, they may wish to export codings into another format, for instance, so that they can apply existing automatic annotation techniques to it. As with importation, possible export formats are too numerous and varied for us to implement converters for them all. Users who supply their own converters will again be able to install them into local copies of the workbench. Printing and postscript output will be available as a function closely allied to display. Stylesheets can be used to produce other output formats which give specific views of the data; for instance, it is possible to use them to construct HTML or tabular information for input into spreadsheets or statistical software. We are still considering how best to export information for visualisation of complete data sets, as required for data mining techniques.

7 Tools for Developers

There are two basic functions which will be required for corpus and coding scheme developers: adding a new coding level, and creating or editing a stylesheet. For adding a new coding module, we intend to support good practice by creating a template for storing information about each type of coding which leaves space for describing working practice, who coded each file, exactly what form of the coding manual was used, and so on. We are still considering what sorts of tools will best facilitate the DTD editing and stylesheet creation essential for new coding schemes, but here the workbench may itself be of service. XSL is an XML language, and current developments in XML suggest that DTDs will soon be written in XML itself using "XML schemata" [13, 14], so that DTD and stylesheet editors could be written quickly using the workbench. Editors written in this way could abstract away from the syntactic details which users find so difficult to deal with, leaving them to concentrate on the structure of the corpus additions.

8 Timescale

Obviously, this is an ambitious project, and the tools for developers will take some time to settle, not least because they require developments to the underlying markup languages. The workbench is scheduled to be released in December 1999. By this time we would expect to provide reasonable support for using the coding schemes which we have implemented. Note that this still leaves coding scheme and tool developers in a better situation than they have been previously, since, given the willingness to learn XML and XSL, software developers will already be able to use the workbench to implement

new coding schemes, coding tools, and display functions. This process should be faster than starting from scratch, especially with the examples implemented in the workbench.

9 Acknowledgements

This work was funded by the European Union as part of the MATE project (Telematics LE4-8370). We are grateful to project participants at our partner sites (NIS, Odense; CSELT, Turin; DFE, Barcelona; DFKI, Saarbrücken; IMS, Stuttgart; ILC, Pisa; and TID, Madrid) and to the MATE Advisory Panel for informing our thinking on these issues.

References

The EAGLES Consortium. EAGLES. http://www.ilc.pi.cnr.it/EAGLES/home.html.

Amy Isard et al. MATE Deliverable 3.1. http://mate.nis.sdu.dk/about/deliverables.html.

Andreas Mengel et al. MATE Query Language. http://www.ims.uni-stuttgart.de/ mengel/.MATE/Specs/quer.html.

Marion Klein et al. MATE Deliverable 1.1. http://mate.nis.sdu.dk/about/deliverables.html.

Giovanni Flammia. The Nb annotation tool. http://www.sls.lcs.mit.edu/ flammia/Nb.html.

Andreas Mengel and Uli Heid. Query language for access to speech corpora. In *Forum Acusticum*, March 1999. (ASA,EAA,DEGA).

Andreas Mengel and Uli Heid. A query language for research in phonetics. In *ICPhS 99 (International Congress of Phonetic Sciences)*, August 1999.

University of Rochester. The DAT system. http://www.cs.rochester.edu/research/trains/ annotation/.

The MATE Project. Mate webpages. http://mate.mip.ou.dk/.

Henry Thompson and David McKelvie. Hyperlink semantics for standoff markup of read-only documents. In *SGML Europe'97*, May 1997.

The W3C. Extensible Markup Language. http://www.w3.org/TR/REC-xml.

The W3C. Extensible Stylesheet Language. http://www.w3.org/TR/WD-xsl.

The W3C. Schema for object-oriented XML. http://www.w3.org/TR/NOTE-SOX.

The W3C. XML Schema requirements. http://www.w3.org/TR/NOTE-xml-schema-req.