The MATE Workbench

Laila DYBKJÆR and Niels Ole BERNSEN
Natural Interactive Systems Laboratory, University of Southern Denmark
Science Park 10, 5230 Odense M, Denmark
laila@nis.sdu.dk, nob@nis.sdu.dk

The aim of the European Telematics project MATE (Multilevel Annotation Tools Engineering) has been to facilitate the use and reuse of spoken language resources by addressing theoretical issues and implementing practical solutions. MATE was launched in 1998 in response to the increasing need - not least in the area of spoken language dialogue systems research and development - for tools and standardisation efforts in support of efficient markup of spoken dialogue corpora at multiple levels.

To provide a solid basis for the coding standard to be proposed by MATE, more than sixty existing coding schemes belonging to five different coding levels (i.e. prosody, (morpho-) syntax, co-reference, dialogue acts, and communication problems) and their cross-level interaction were reviewed (Klein et al. 1998). On this background, the MATE markup framework (Dybkjær et al. 1998) was proposed as a standard which can facilitate uniform description of schemes across levels. For each level, one or several of the reviewed coding schemes were adopted as starting-points for the definition, following the MATE markup framework, of best practice coding schemes for implementation in the MATE workbench (Mengel et al. 2000).

The MATE software workbench (Isard et al. 1998, Isard et al. 2000, Dybkjær and Bernsen 2000) supports the MATE markup framework and incorporates the MATE best practice coding schemes. It is implemented in Java 1.2. It has been tested on Unix (Solaris) and Windows (NT and 98) but should run on any platform for which Java 1.2 is available. The workbench has a modular architecture which facilitates the addition of new modules and new tool functionality by its users. XML is used for internal file representation.

The MATE workbench has the following major components: an internal database which is an in-memory representation of a set of hyperlinked XML documents; a query language and processor which are used to select parts of this database for subsequent display or processing; a stylesheet language and processor which respectively define and implement a language for describing structural transformations on the database; a display processor which handles the display and editing actions; and a user interface which handles file manipulation and tool invocation.

Stylesheets are used for specifying the visual presentation of data to users. Stylesheets are written in the MATE Stylesheet Language (MSL). The emerging standard in this area is XSLT, but XSLT was not fully defined when the workbench was being designed, and lacked various necessary functionalities. It was therefore decided to implement MSL which uses the MATE query language but is otherwise similar to XSLT.

The two main functionalities of the MATE workbench are support for annotation of spoken dialogue corpora and similar linguistic material, and information extraction from annotated corpora. The annotation support includes:

- Ready-to-use best practice coding schemes for the levels mentioned above. Stylesheets are included which allow the user to view and annotate files in an appropriate way using any of the implemented best practice schemes. Example dialogues are also provided which illustrate how a tagged dialogue and the accompanying tag set will be shown when a particular best practice scheme is being applied. Internally in the workbench, an annotated dialogue consists of a set of references to the transcription of the dialogue and possibly to other coding files. By default, the transcription is assumed to refer to timeline information.
- There is no example coding module for transcription in the MATE workbench.

Instead, a converter from Transcriber format (http://www.etca.fr/CTA/gip/Projets/Transcriber/) to MATE format enables transcriptions made using Transcriber to be annotated using the MATE workbench.

- Easy addition of new coding schemes for existing or new coding levels. New coding schemes are entered by using the coding module editor. A coding module is the standard type of coding scheme description proposed by MATE. A coding module prescribes what constitutes a coding, including markup representation and relations to other codings.
- Editing of files. MATE operates with three basic kinds of files. XML files are coding files. MSL files are stylesheets used for visualisation of codings and for annotation support. Runfiles specify which stylesheet to apply to which XML file. Any XML, MSL or runfile in a MATE project folder can be opened and edited.
- Creation of new project folders and addition or creation of new files, including header file documentation.
- An audio tool for listening to speech files and displaying the sound file as a waveform. The audio tool might be used during transcription if a user has added an appropriate transcription coding module. As it stands, the audio tool is intended to act as a support tool during annotation and annotation review. For instance, when annotating communication problems there is often a need for listening to the speech file in order to disambiguate an utterance and diagnose what went wrong.
- Conversion to XML format of files in other formats. For the moment, conversion from XLabels and BAS Partitur to XML is enabled. New converters can easily be added, including converters which export from XML to other formats, such as HTML.

As regards information extraction, a powerful query tool is available for selecting the document(s) to be searched and for specifying the information to be subjected to a query. The information which can be extracted includes statistical information.

An online help facility is available and may be consulted at any time during use of the MATE workbench.

The workbench is continuously being improved. We believe that the coding module editor shows that a walk-up-and-use interface for non-XML-literate users can be created. A major current drawback of the workbench interface is the use of stylesheets for visualisation. Writing stylesheets is cumbersome and definitely not something users should be asked to do in order to define how codings based on their new coding module should be displayed. Therefore, it is high on our wish-list to create a better user interface for defining visualisations.

Acknowledgements

We gratefully acknowledge the support of MATE provided by the European Commission’s Telematics/Language Engineering Programme. We would also like to thank all MATE partners. Without the very considerable joint efforts of the project consortium it would not have been possible to build the MATE Workbench.

References

MATE deliverables are available from the MATE web site at http://mate.nis.sdu.dk.

Dybkjær, L. and Bernsen, N. O. (2000) *The MATE Workbench.* Proceedings of the LREC’2000 workshop on Data Architectures and Software Support for Large Corpora, Athens, 33-37.

Dybkjær, L., Bernsen, N. O., Dybkjær, H., McKelvie, D. and Mengel, A. (1998) The MATE Markup Framework. MATE Deliverable D1.2.

Isard, A., McKelvie, D., Cappelli, B., Dybkjær, L., Evert, S., Fitschen, A., Heid, U., Kipp, M., Klein, M., Mengel, A., Møller, M. B. and Reithinger, N. (1998) Specification of Workbench Architecture. MATE Deliverable D3.1.

Isard, A., McKelvie, D., Mengel, A., Møller, M. B., Grosse, M. and Olsen, M. V. (2000) Data Structures and APIs for the MATE Workbench. MATE Deliverable D3.2.

Klein, M., Bernsen, N. O., Davies, S., Dybkjær, L., Garrido, J., Kasch, H., Mengel, A., Pirrelli, V., Poesio, M., Quazza, S. and Soria, C. (1998) Supported Coding Schemes. MATE Deliverable D1.1.

Mengel, A., Dybkjær, L., Garrido, J., Heid, U., Klein, M., Pirrelli, V., Poesio, M., Quazza, S., Schiffrin, A. and Soria, C. (2000) MATE Dialogue Annotation Guidelines. MATE Deliverable D2.1.