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, nob}@nis.sdu.dk

Abstract

The growing commercialisation and sophistication of spoken dialogue systems has increased the need for methods and tools in support of annotating and extracting information from spoken dialogue resources, unimodal as well as multimodal. This paper describes results and future prospects of the MATE project which investigated spoken dialogue annotation and built a workbench in support of annotation and information extraction. This workbench may have the potential for being generalised to providing tools support for multimodal dialogue annotation.

1. Introduction

The growing commercialisation and sophistication of spoken dialogue systems has generated a strong need for spoken dialogue corpus annotation and re-use of the annotated resources. The next generation of dialogue systems most of which are still in the research labs, will not be speech-only systems but will include other modalities as well, such as gestural input and graphics output. This forces extension of the focus of corpus annotation to multimodal spoken dialogue resources. During the next few years, we will probably be seeing a series of initiatives in tools support, annotation schemes and standards for multimodal dialogue corpora.

This paper describes results and future prospects of the MATE project (http://mate.nis.sdu.dk) which ended by the end of 1999. MATE has successfully addressed some core issues in spoken dialogue annotation and tools support and may have the potential for being generalised to providing tools support for multimodal dialogue annotation. This is now being investigated in the ISLE project which started in 2000.

2. MATE

MATE was launched in March 1998 in response to the increasing need for standards and tools in support of creating, annotating, evaluating and exploiting spoken language resources. Corpus annotation is time- and cost-intensive, and re-use of annotated data would seem very attractive. So far, however, re-use has usually been difficult and time-consuming if not downright unattractive due to the lack of standards and widely used tools. MATE aimed to facilitate re-use of spoken language resources by addressing theoretical issues as well as the practical implementation of solutions.

MATE reviewed more than 60 existing annotation schemes relating to the annotation levels addressed in the project, i.e. prosody, (morpho-)syntax, co-reference, dialogue acts, communication problems, and cross-level issues. The resulting report, (Klein et al. 1998) provides details on their coding book, number of annotators who have worked with it, number of annotated dialogues/-segments/utterances, evaluation results, underlying task, list of annotated phenomena, and markup language used. Annotation examples are also provided. The amount of pre-existing work varies enormously from level to level, which causes very different state-of-the-art problems at the individual levels. The quality of descriptions of the individual coding schemes analysed, varies considerably which made it extremely difficult to compare schemes even within the same annotation level. Furthermore, with such non-standardised descriptions it would be impossible to create generally reusable tools within, as well as across, levels.

The collected information and joint experience in the consortium formed the basis for the development of the MATE markup framework for spoken dialogue corpus annotation at multiple levels, including those mentioned above (Dybkjær et al. 1998). The core concept of the framework is the *coding module* which extends and formalises the concept of a coding scheme. Roughly speaking, a coding module describes everything that is needed in order to perform a certain kind of markup of a particular spoken language corpus. A coding module prescribes what constitutes a coding, including markup representation and relations to other codings.

The MATE markup framework ensures a common approach and uniform description across levels. For each annotation level, one or more state-of-the-art coding schemes were selected to form the basis of the best practice coding schemes proposed by MATE (Mengel et al. 2000). Common to the selected coding schemes is that these are among the most widely used schemes for their level, which means that they have been used by several annotators and for the annotation of many dialogues. Examples of coding schemes selected for the MATE workbench are SAMPA, ToBI, MapTask, Verbmobil and MUC-7. All MATE best practice coding schemes are expressed in terms of coding modules.

A range of existing tools for annotating spoken dialogues were reviewed early in the project to provide input to the specification of the MATE workbench (Isard et al. 1998). Examples of reviewed tools are the Alembic workbench, AnnoTag, DAT and Nb.

Building on the specification, the MATE markup framework, and the best practice coding schemes, a basic set of coding functionalities has been implemented in the Java-based MATE workbench. Implementation details are described in (Isard et al. 2000). Workbench usability is supported by the MATE markup framework which facilitates use of the same set of software tools and the same interface look-and-feel independently of annotation level, and serves as an intermediate layer ensuring that the user interface need

not change even if the underlying XML coding file representation might be changed and vice versa.

The functionalities offered to the user by the MATE workbench include:

- The best practice coding modules mentioned above.
- The possibility of adding new coding modules via an easy-to-use interface.
- Support for header file documentation.
- Support for annotation/transcription according to some selected coding module.
- An audio tool for listening to speech files and displaying the sound file as a waveform.
- A number of default style sheets which define how output to the user is visually presented.
- Information extraction from annotated MATE corpora, including statistical information.
- Import of files from XLabels and BAS Partitur to XML format.
- Easy addition of other converters (import as well as export).

The following sections describe and illustrate the MATE workbench functionalities in more detail and discuss future prospects.

3. Workbench functionalities

The workbench is written in Java 1.2 to make it platform-independent. It has been tested on Unix (Solaris) and Windows (NT and 98). It does not currently run on Macintoshes because there is not yet a version of JDK1.2 for this platform. The MATE workbench allows reuse of other software, either by using XML as data interchange format or by calling other Java modules which access the internal representation and the MATE display structure using well-defined interfaces.

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.

To run the MATE workbench on a Windows'98 PC the user opens a DOS prompt in the directory with the MateWorkbench.jar file and types 'java mate.Workbench' to open the windows shown in Figures 1 and 2.

The 'controller window' is the main window from which all workbench tool windows can be opened. New tools can be added to the workbench and will then automatically be made available in the tools menu. One or more corpus folder windows can be opened from the 'File' menu. Status messages are written into the white area under the menu. 'Corpus folder' windows are used for browsing, adding or editing corpus files. The 'help' menu gives access to the 'help window' shown in Figure 3, which explains and describes various workbench topics.

The MATE best practice coding schemes are included in the workbench as example coding modules which the user may choose to use. These allow the user

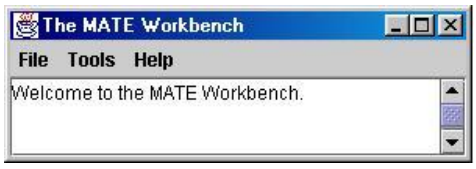

Figure 1. The MATE controller window.

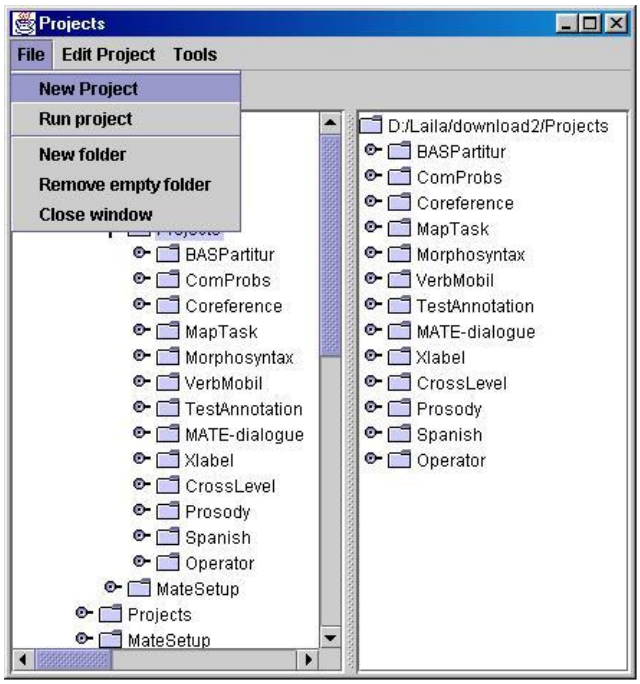

Figure 2. The corpus folder window.

Figure 3. The MATE help window.

to annotate corpora at the levels addressed by MATE. However, users are also offered the possibility of

adding new coding modules for existing or new levels via the easy-to-use interface of the MATE coding module editor. This editor is accessible from the 'tools' menu in the 'controller' window, cf. Figure 4. The editor has a graphical user interface which resembles the user interfaces found in most popular and widespread programs. It contains features like 'copy and paste' and 'drag and drop' which make it easy to reuse parts of one coding module in another. The markup declaration section of the coding module is represented as a tree, and the user adds entities, elements, attributes, and comments to the tree to construct the markup declaration. For each node the name, type, etc. is specified. The tree can be parsed to create a coding module text document. On the basis of the entered markup declaration, a DTD is automatically generated that defines which tags are available and how they can be used during markup of a corpus. The interface of the coding module editor builds on the MATE markup framework.

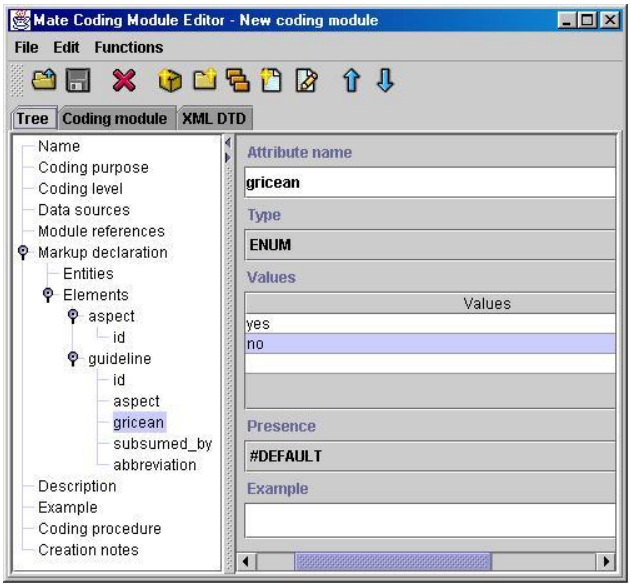

Figure 4. The coding module editor.

From the 'file' menu in the corpus folder window (Figure 2) it is possible, e.g., to create a new project folder. The 'edit project' menu allows the user to create a new file in a folder, add an existing file, or remove a file. If the user chooses to make a new file, s/he is first asked what kind of file s/he wants to create (XML, stylesheet (msl), or run). If the choice is an XML file, the window shown in Figure 5 will appear. The information to be filled into this window is a kind of header information plus administrative information. The new file is given a name, the user can indicate a stylesheet for use during annotation and files to which the new file will refer, such as a transcription. A coding module or a DTD must be selected which will form the basis for the annotation and determine which tags can be used. In addition, as Figure 5 shows, some information about the coder and the file may be included. If 'add' is chosen from the 'edit project' menu the window show in Figure 6 will appear. The user can browse folders and files on the computer and select the file to be added.

One of the tools available from the tools menu in the corpus folder window is the conversion tools which enables the user to easily convert files in BasPartitur or

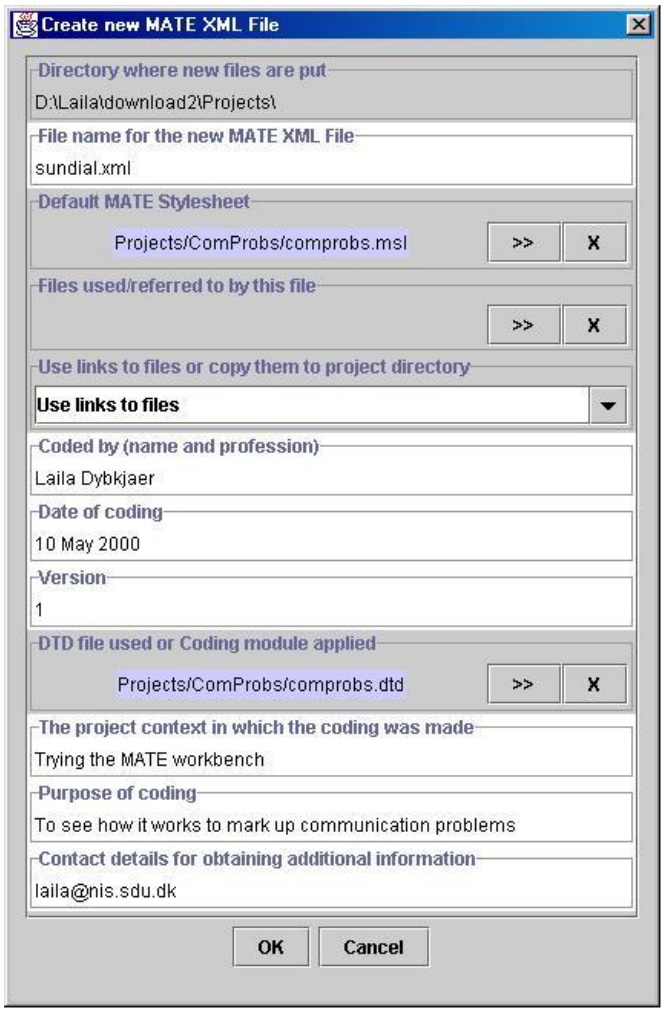

Figure 5. Creating a new XML file.

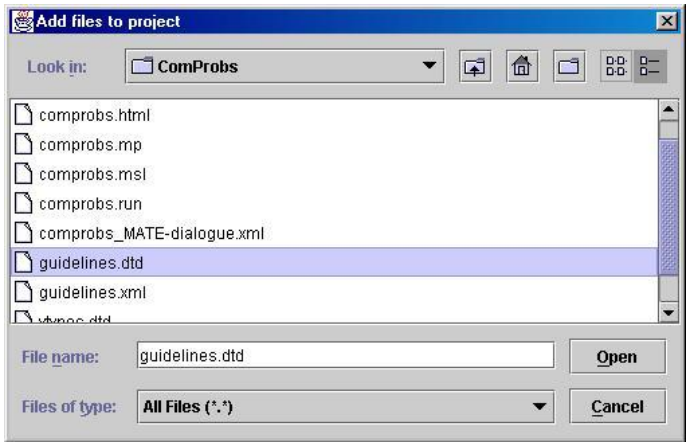

Figure 6. Adding a file to a project folder.

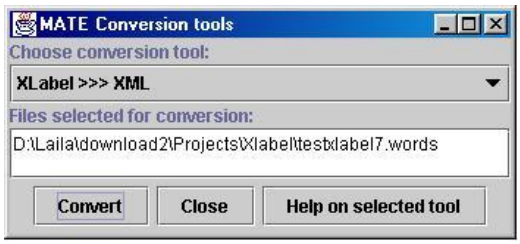

Figure 7. Converting from Xlabel format to XML.

Xlabel format to XML. The conversion tool window for Xlabel format is shown in Figure 7. The user can brow-

se and select the file to convert into XML. Other converters can easily be added to the workbench. Export to file formats other than XML may be achieved by using style sheets. For example, information extracted by the query tool may be exported to HTML in order to serve as input to a browser.

Support for transcription in the workbench itself is fairly primitive. However, it is possible to use Transcriber (www.etca.fr/CTA/gip/Projets/Transcriber) for the transcription process and then use the transcription files in the MATE workbench for further annotation at the desired level(s). Figures 8 and 9 show two examples of what the interface looks like when some of the MATE best practice coding modules are used for annotating dialogues. Figure 8 shows an example of annotation at the morpho-syntactic level (word level). Figure 9 shows dialogue acts annotation using MapTask coding.

A number of default style sheets define how output is visually presented to the user. For instance, phenomena of interest in the corpus may be given a certain colour or shown in boldface, cf. Figures 8 and 9. A MATE stylesheet is written in the MATE Stylesheet Language (MSL). The emerging standard in this area is XSLT. Since XSLT was not fully defined when the workbench was being designed, and since XSLT lacks various necessary functionalities it was decided to implement a slightly different and simpler transformation language. MSL uses the MATE query language but is otherwise similar to XSLT. The user may modify a style sheet or define new ones. However, as no stylesheet editor is available yet, a fairly detailed understanding of XSLT concepts and structure is required.

The audio tool, cf. Figure 11, allows the user to load a sound file which is then displayed as a waveform and which can be played. It is also possible to play parts of a file by setting up display actions with a stylesheet. Clicking on the green PLAY buttons in Figure 9 will cause the utterance audio file to be played.

The workbench supports information extraction from annotated corpora, including statistical information. Moreover, computation of important reliability measures, such as kappa values, is enabled. Figure 10 shows the MATE query window. The user must first select the documents to be queried. Secondly, one chooses element types to be included in the query expression from those available in the selected documents. Then the query expression can be built. The result of a query is a document with a list of tuples of elements that are hrefs to the elements found. Since the output of the query is XML, the results can be displayed to the user in the same way as the data itself, using a stylesheet. A default stylesheet is provided, but different views could be desirable for specific purposes.

4. Future prospects

A major aim of MATE has been to support spoken language dialogue systems development. The market for these systems is growing rapidly and so is the need for re-usable resources and tools which facilitate their creation. MATE has therefore in many respects been timely and appropriate in responding to actual needs.

However, the next generation of spoken dialogue systems are taking first steps towards more natural interactivity by combining speech with other modalities such as gesture and facial expression. As such multimodal dialogue systems are gaining ground, the need for reusable multimodal resources, for tools, and for standardisation efforts in support of the development of such systems is also increasing. A natural follow-up on MATE would therefore be to map out existing coding schemes and tools for markup of gesture or facial expression using the same successful approach as for spoken dialogue in MATE. Several projects in the area have been initiated recently, e.g. TalkBank and ATLAS. A survey of such coding schemes and tools is being made in the ISLE project. ISLE results plus MATE results could form an appropriate point of departure for the development of a standard framework covering not only speech but also gesture and facial expression, and for the development of best practice coding schemes and a set of tools in support of annotation and exploitation of multimodal dialogues.

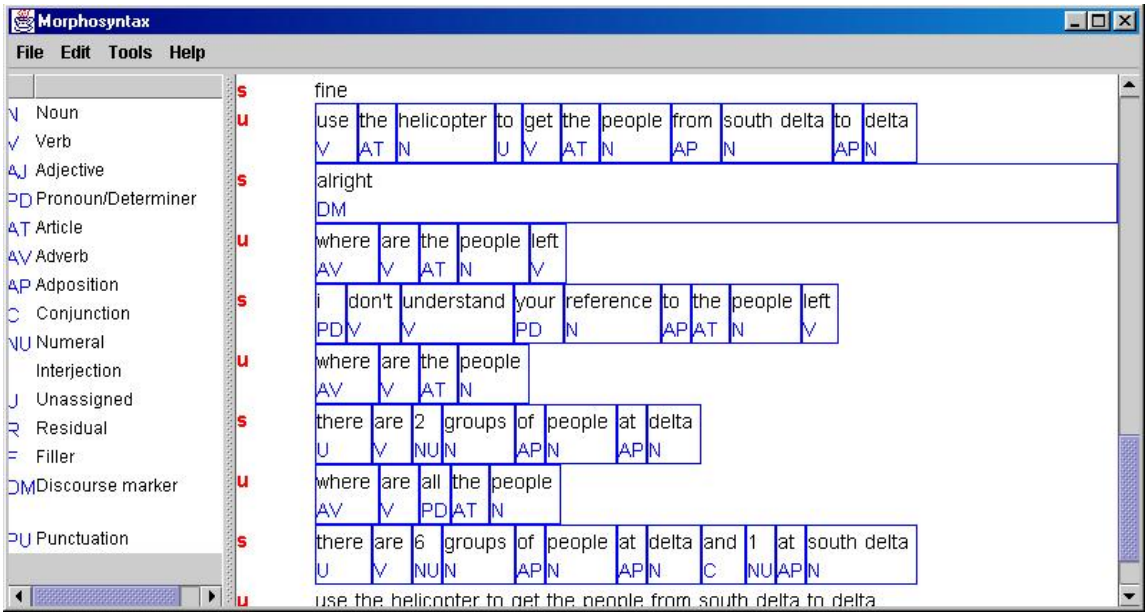

Figure 8. Dialogue annotation at the morpho-syntactic level (word level).

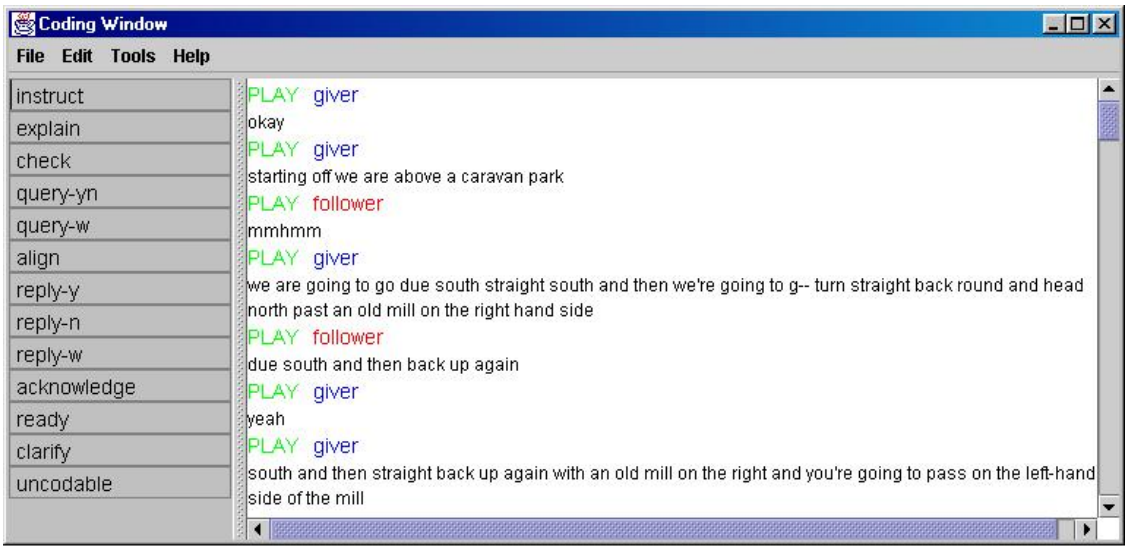

Figure 9. Dialogue annotation of speech acts.

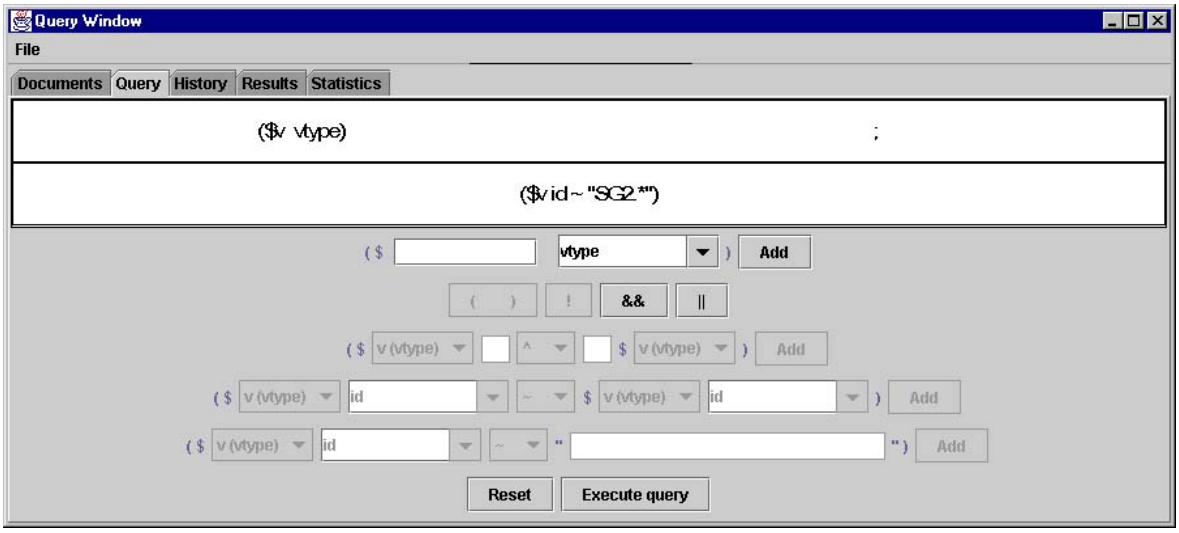

Figure 10. The MATE query window.

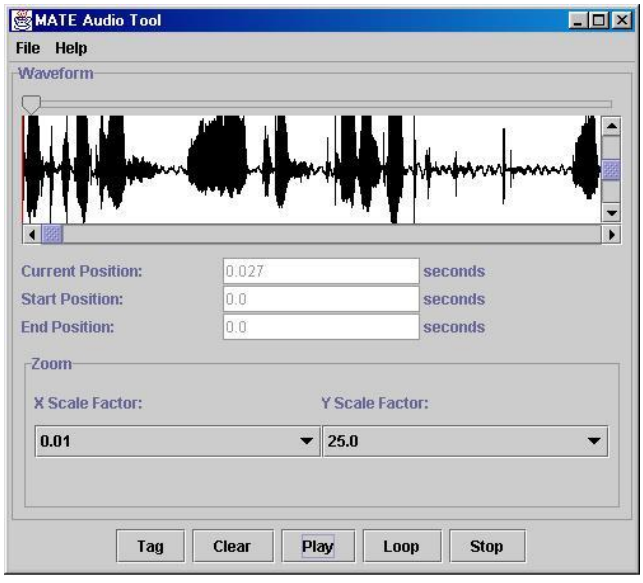

Figure 11. The audio tool.

5. References

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

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, S., 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.