An Experiment in Unifying Audio-Visual and Textual Infrastructures for Language Processing Research and Development

Kalina Bontcheva[†], Hennie Brugman[‡], Albert Russel[‡], Peter Wittenburg[‡], Hamish Cunningham[†]

[†] Department of Computer Science, University of Sheffield, Sheffield, UK

`<kalina,hamish>@dcs.shef.ac.uk`

[‡] Max-Plank Institute for Psycholinguistics, Nijmegen, The Netherlands

`<firstname.lastname>@mpi.nl`

Abstract

This paper describes an experimental integration of two infrastructures (Eudico and GATE) which were developed independently of each other; for different media (video/speech vs. text) and applications. The integration resulted into gaining an in-depth understanding of the functionality and operation of each of the two systems in isolation, and the benefits of their combined use. It also highlighted some issues (e.g., distributed access) which need to be addressed in future work. The experiment also showed clearly the advantages of modularity and generality adopted in both systems.

1 Introduction

This paper describes the integration of two infrastructures (Eudico and GATE) which were developed independently of each other; and for different media (video/speech vs. text) and applications. Such integration was needed in order to have an application where the end users (linguists and language engineers) who annotate video and speech corpora with textual transcriptions in Eudico, can also benefit from language processing tools and viewers from GATE.

Eudico (European Distributed Corpora) is a distributed multimedia infrastructure supporting creation, presentation and analysis of annotations of speech and video corpora (Brugman et al., 1998). Annotations of all kinds of user-definable types can be time-aligned with the speech/video data so that dynamic and simultaneous viewing is possible. For example, when the user sets a new media time this time is reflected in all annotation viewers, and, vice versa, when the user selects an annotation this time selection is shown in all viewers, including the media player: viewing of media and transcription data is synchronised.

GATE (General Architecture for Text Engineering) (Cunningham et al., 1997; Cunningham et al., 1999) is an architecture, framework, and development environment, providing representation and storage of language data together with infrastructural support for building and deploying language engineering applications. Its plug-and-play support for processing modules and data viewers lowers the overhead of building such applications and facilitates code-reuse.

The idea behind this experiment is to combine and build on the strengths of these two architectures, thus bridging the gap between transcriptions of speech and video, and language engineering tools. From the user's perspective, this entails simultaneous manipulation of media, transcriptions and linguistic data in a uniform and synchronised way. The integration with GATE provides Eudico applications with ways to represent, display and manipulate linguistic data and, more importantly, to run GATE language processing modules on the text transcriptions (e.g., part-of-speech tagger, name-entity recogniser). In this way, linguists and language engineers developing speech/video corpora are assisted in the corpus annotation task by language processing tools and viewers.

The main question that had to be answered is whether it is possible to integrate into one application two separate architectures. The major difference between the two comes from the media structuring: speech/video annotation in Eudico is time-based while text annotation in GATE is offset-based. Therefore, we had to find a way of storing and accessing time information for offset-based linguistic objects. From an implementational viewpoint, this entailed:

1. Mapping objects from the GATE world to objects in the Eudico world in such a way that Eudico's views give a meaningful representation of GATE data, while keeping their dynamic and synchronised nature.
2. Embedding GATE viewers in Eudico viewers and making them time aware with minimal rewriting of existing code.

The next three sections are devoted to discussing the design and implementational aspects of these two problems. Section 2 describes how the actual mapping of objects between the architectures is done. Section 3 describes the synchronisation mechanism allowing GATE and Eudico viewers to operate and update in parallel with the media being played. Section 4 describes the specific GATE viewers used in this project. The pilot application is described in section 5. Section 6 concludes this report by summarising the outcome of this work and pointing to a set of open issues.

2 Integrating the Two Data Models

2.1 Eudico Data Model

The key concepts underlying Eudico's data model are:

Corpus - a collection of Transcriptions (or of subcorpora).

Transcription - all annotations that refer to one media file, or that describe one uninterrupted recorded event.

Tier - a collection of Tags that are strictly consecutive in time and that annotate one specific phenomenon. Tiers can also have meta information (e.g. name, transcriber).

Tag - a typed set of values applying to a time interval. A Tag is part of exactly one Tier. Tags on the same Tier may not overlap in time. Tags are either explicitly linked to media time or ordered in time.

These concepts are part of Eudico's Abstract Corpus Model (ACM). The ACM was designed to abstract the specifics of corpus annotation formats from the tools that work on the annotation data, thus making Eudico corpus format independent.

2.2 GATE Data Model

GATE's data model is based on the TIPSTER architecture (Grishman and Sundheim, 1996). The main classes are:

Collection - a set of Documents which can be loaded, stored, and processed together.

Document - consists of a document content (e.g. text) and a set of Annotation objects associated to the document content by means of spans (objects specifying the begin and end offset of the annotation).

Annotation - has a set of spans (specify the text parts covered by this annotation), type (e.g. part-of-speech (POS)), and a set of Attribute objects which hold further information about the annotation (e.g. time=12345, category="Noun").

Attribute - a feature-value pair which can hold any type of data as a value. Attributes can be associated to any of the above classes - collection (e.g., creator), document (e.g., language, media type), and annotation (see above).

2.3 Mapping between the two worlds

There is a two-way mapping between Eudico and GATE objects since GATE objects are constructed from Eudico ones (with a special 'Eudico-to-Gate'tool) that were initialised from another existing corpus, and, vice versa, Eudico objects are created from GATE ones when a stored Gate collection is loaded in Eudico for further processing. This mapping is realised by implementing the proper parts of the Eudico's ACM using GATE's API and data structures. We chose to specify the mapping only for a set of data that can be treated in a meaningful way in both environments, namely transcribed speech utterances.

Eudico corpora are mapped to GATE collections. All transcribed utterances on a given media file are ordered according to their place in the original file and form a *Document*. All the information in the document is accessed through time attributes and annotation spans. In this way, only the relevant document part(s) are manipulated at each given time point. All *Tag*s from the *Tier*s will are added as *Annotations* of type `utterance`. *Tier* information is encoded as an *Attribute* on the `utterance` annotations created for the *Tag*s. *Tier* objects can be re-created by selecting all `utterance` annotations which have the same value for the attribute `tier` (see the example below). In order to provide a two way link between *Tag*s and `utterance` annotations, all Tags have associated span information and all `utterance`s have associated time information.

Based on the `utterance` annotations, which provide a link to Eudico's time-based world, new linguistic data in the form of other annotations can be added now to the media corpus.

We experimented with two types of such data:

- `POS` - part of speech annotations which are currently obtained from hand-annotation.
- `syntaxTree` - syntax trees obtained by manual annotation.

2.4 Example

The following example of the mapping is based on an example of a time-aligned file from the CHILDES Corpus (MacWhinney, 1999).

```
@Begin
@Filename: boys73.cha
@Participants
ROS Ross Child, MAR Mark Child,
FAT Brian Father, MOT Mary Mother
*ROS: yahoo.
%snd: "boys73a.aiff" 7349 8338
*FAT:   you got a lot more to do # don't you?
%snd: "boys73a.aiff" 8607 9999
*MAR: yeah.
%snd: "boys73a.aiff" 10482 10839
*MAR:   because I'm not ready to
        go to <the bathroom>[>] +/.
%snd: "boys73a.aiff" 11621 13784
```

This file is first parsed and Eudico Tiers and Tags are created as follows:

```
Tiers: ROS, MAR, FAT, MOT.
Tags for ROS: {(7349, 8338, "yahoo")}
Tags for FAT: {(8607, 9999,
     "you got a lot more to do# don't you?"}
Tags for MAR: {(10482, 10839, "yeah"),
              (11621, 13784, "because..+/.")}
```

Speech utterances are combined into a GATE document:

```
yahoo. You got a lot more to do# don't you?
0...|5...|10..|15..|20..|25..|30..|35..|40..

yeah. because I'm not ready to go to...
45..|50..|55..|60..|65..|70..|75..|80..
```

Utterance annotations are created for each Tag:

Id	Type	Span start	Span end	Attribute
1	utterance	0	5	Tier=ROS, Time=(7349,8338)
2	utterance	7	43	Tier=FAT, Time=(8607,9999)
3	utterance	45	49	Tier=MAR, Time=(10482,10839)
4	utterance	51	102	Tier=MAR, Time=(11621,13784)

3 Time-based Synchronisation

3.1 Encoding Time Information for Linguistic Objects

Eudico objects can be time-aligned which provides a way of synchronising all Eudico viewers by always showing the information relevant to the current point of time in the media. Therefore it is desirable for GATE linguistic objects to be time-aligned as well when such information is available.

This is achieved by encoding the time in milliseconds as a value of an *Attribute* object, which is then associated with linguistic *Annotation* objects (e.g. `POS` annotations). In this way, annotations with such an attribute are linked to the media time and can be manipulated and displayed in the same way as Eudico objects.

For example, given a transcription tier for one speaker, `POS` annotations are added using GATE. Each `POS` annotation has to be associated with both a time interval and a text span.

The procedure is as follows:

- Use GATE to add `POS` Annotations
- Display the `POS` annotations in a GATE viewer
- Select an Annotation in this viewer (for multi-span annotations, select one of the spans)
- Associate the span of this annotation with a time interval that is taken from a time selection set with the Eudico media player.

3.2 Using Time Information to Synchronise GATE Viewers

Eudico has a time-handling mechanism where, at the time of creation of a new viewer, all relevant time points are registered with the media player. These time points are derived from the time information of the annotation objects that are to be displayed in the viewer. During media playback an event is generated for each timepoint and based on these events, the viewers update themselves to reflect the current time in their own specific way. This mechanism was extended to the domain of GATE viewers in a non-trivial way. Extra timepoints are taken from the attributes of time-aligned annotations and also registered with the media player. Additional events are generated at play back time and passed on by the Eudico viewer to the proper embedded GATE viewer. These events can then be used by the GATE viewer to show/highlight annotations at the appropriate time, resulting in time synchronisation between all Eudico and GATE viewers. Examples are given below.

4 Re-using GATE Viewers inside the Eudico Interface

4.1 Disguising GATE Viewers as "Native" Eudico GUI classes

In order to provide creation, editing, and visualisation of linguistic data, we embedded GATE GUI modules into classes implementing Eudico's interface for a panel displaying a single Tag. In this way, the various Eudico viewers can manipulate them in the same way as the "native" Eudico ones. The wrapper classes also provide the time synchronisation functionality described in the previous section. We experimented with two GATE viewers - POS and SyntaxTree viewers - corresponding to the chosen linguistic annotations.

The POS viewer works at the level of orthographic transcription because that is a type of text that can be handled meaningfully in both the GATE and Eudico domains. Utterances at this level are almost all already time-aligned which means that the POS viewer can operate sensibly even if no finer time alignment exists. Words in the utterances can be selected for manual annotation with part of speech data (the annotation dialog is shown in Figure 1). These annotations are then visualised by textually aligning them with the proper span of the speech utterance.

The TreeViewer displays an utterance together with one or more (partial) syntax trees of sentences within this utterance (see Figure 2). The annota-

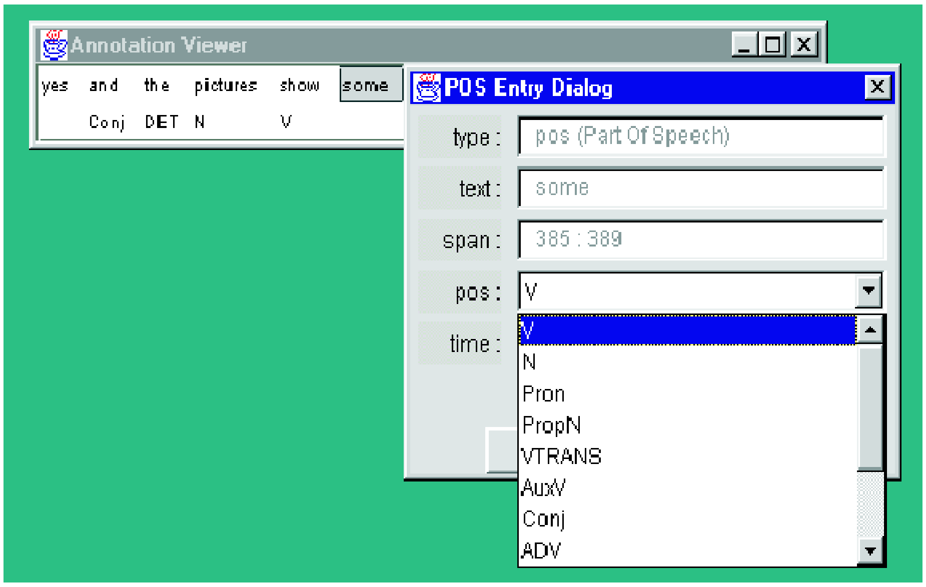

Figure 1: The Part-Of-Speech (POS) viewer

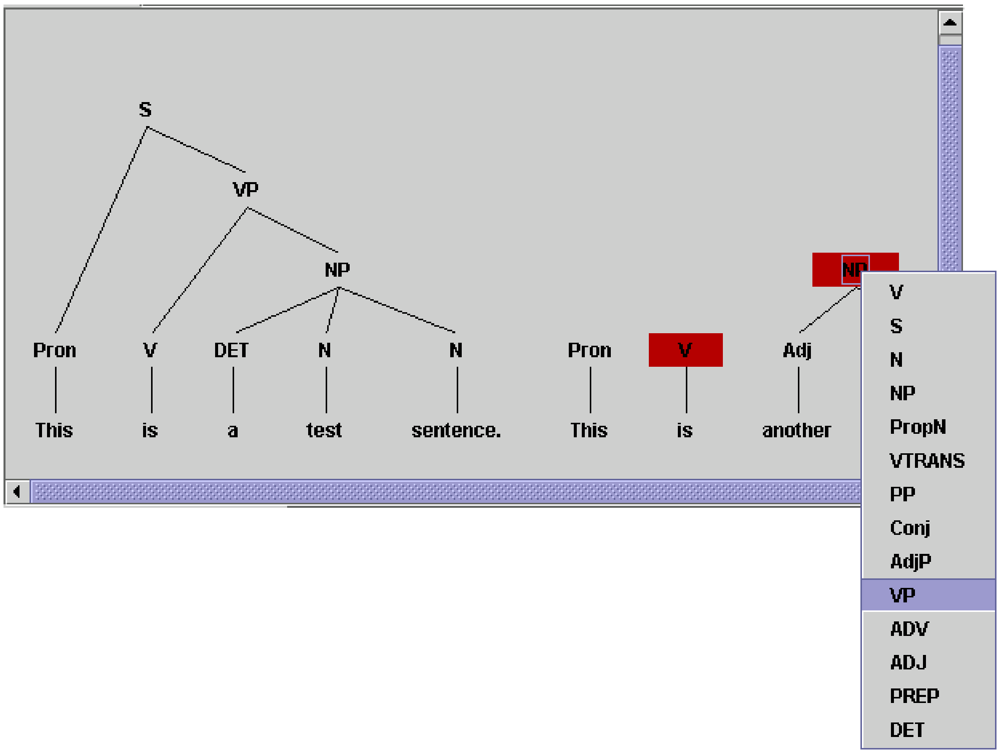

Figure 2: The TreeViewer

tion process starts from the words and proceeds upwards. The appropriate category can be selected from a menu of availabe categories which is constructed from the corresponding stereotype (see following section). Non-terminal nodes can be combined into higher-level non-terminals (e.g. V and NP into a VP) by selecting all relevant nodes for the new category and selecting the category itself from the menu. Deletion of selected nodes is avaialble too.

Similar to all GATE viewers, both the POS and Syntax Tree viewers are implemented as Java Beans which enables their easy reuse and configuration. On-line demonstrations of GATE applications embedding these viewers are available at `http://www.gate.ac.uk`[1].

4.2 Configuring Viewers for Types of Annotations

Annotations in GATE are flexible and unconstrained data structures. This has the advantage that the architecture is theory-neutral, and can represent a wide range of data. The disadvantage is that the application using them needs meta-information about annotations in order to allow proper creation, editing, and visualisation. Therefore, annotation stereotypes are used for encoding this information, which is then used for configuring the behaviour of the GATE viewers and editors.

These stereotypes (AnnotationStereotype class) have

- annotation type (e.g. POS, SyntTree)
- annotation structure type (single span, multiple span, tree or graph)
- a set of attribute stereotypes specifying the name of the attribute feature, the type of permissible values (e.g., String), and a list of permissible values where applicable.

For example, the stereotype of POS annotations is given in Table 4.1.

Viewers use stereotypes to define what annotation types they can visualise. For example, the syntax tree viewer can register itself as supporting annotations with structure type tree, or even more specifically, annotations of type SyntaxTree. Creation/editing of annotations also uses stereotypes to control what attributes and values can be entered. For instance, in the POSEntryDialog in Figure 1 the list of permissible categories is taken from the permissible values for the attribute `cat` as defined in the POS stereotype (see Table 4.1 for some example values)[2].

5 The Showcase Application

The publicly-available showcase application[3] currently demonstrates the time syncronisation and viewing of linguistic data provided by the integration of the two architectures. The application uses sound media annotated with utterances for each speaker[4]. The data is taken from the ESF corpus[5] and imported via Eudico into a GATE document with annotations. Afterwards some utterances were manually annotated with part-of-speech information using the GATE POS Viewer (see the figure above). The result is stored using GATE's persistence mechanism and is read every time the application runs.

The showcase application allows playback and viewing of media and associated linguistic data. It demonstrates the synchronysed operation of GATE's linguistic viewers embedded into Eudico's media and annotation viewers. The screen shot below shows two types of Eudico viewers used - subtitle and tag list viewer[6]. The subtitle viewer (the window with liela11p.wav caption) shows the current utterance for each speaker (in this case, INN and SLA) and controls the media playback. The other two Eudico viewers (the windows with GATE List View captions) are tag list viewers, one for each speaker, and integrate many instances of the GATE's POS viewer. Each instance displays a particular utterance and all POS annotations related to it.

The embedding Eudico viewers take care of layout, scrolling, update and time synchronisation of the embedded GATE components. They obtain the relevant time data from each GATE POS viewer's annotation data and register it with Eudico's time synchronisation mechanism. They also receive all time events and direct them to the appropriate GATE viewer which then highlights the word which is currently played as a sound by the media player.

In the example screen shot in Figure 3, the current word is **day**. A few milliseconds later, another time event would cause the highlight to be moved to the word **and**, which is the next word to be spoken. When no POS data is available (as for the word **this**), the last word remains highlighted until the next time-aligned annotation becomes current or the end of the utterance is reached. The Eudico viewers also take care of displaying a moving red bar in front of the currently playing utterance.

In addition to this synchronous playing behaviour, time synchronisation is exploited in some other ways: it is possible to select an utterance in one of the tag list viewers. This (time) selection is then reflected in each of the other viewers (by means of a blue bar in front of the overlapping utterances). The media time is automatically reset to the begin time of the selection. Manipulating the media time by dragging the media player's slider is reflected by the red bar and text highlight in the other viewers.

6 Conclusion

This paper described the integration of two infrastructures (Eudico and GATE) which were developed independently of each other; for different me-

[1] The demos require a browser (e.g., Netscape, Internet Explorer) with enabled Java and Java applets.

[2] Eudico's Abstract Corpus Model includes comparable concepts, but no effort to integrate at this level was made in the scope of the pilot project.

[3] Available for download under demos at `http://www.gate.ac.uk`.

[4] Video data could have been used just as easily, but at the cost of having the user install the Java Media Framework (JMF) on her system

[5] http://www.mpi.nl/world/tg/lapp/esf/esf.html

[6] For examples and a discussion of all Eudico viewers see http://www.mpi.nl/world/tg/lapp/eudico/eudico.html

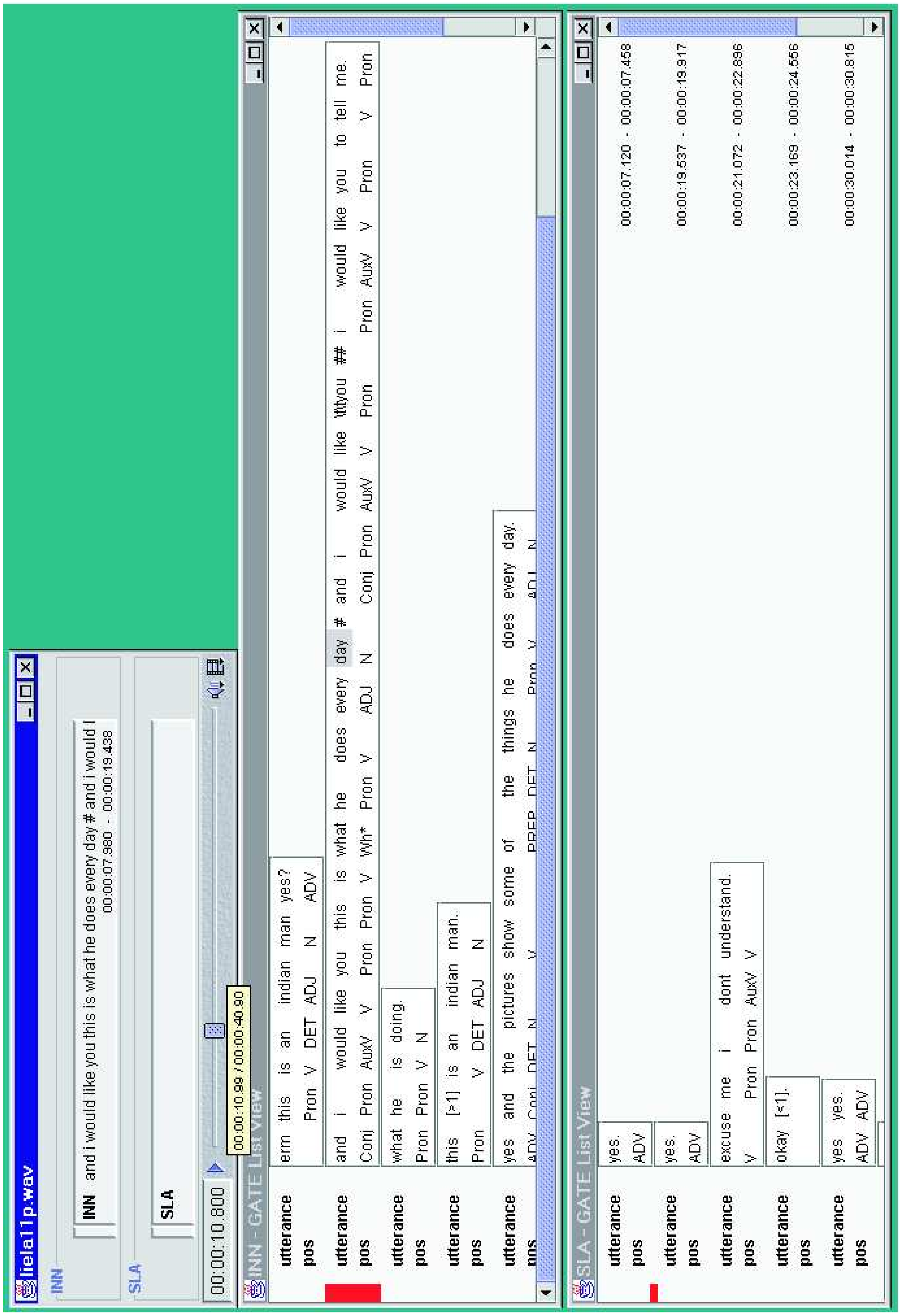

Figure 3: A screen shot of the demo

annotationType	`POS`		
AnnotationStructureType	Single span		
AttributeStereotypes	cat	String	det, n, adj, v, prep, conj, aux
	Time	Long	

Table 1: Stereotype for `POS` annotations

dia (video/speech vs. text) and applications. Such integration was needed in order to have an application where the end users (linguists and language engineers) who annotate video and speech corpora with textual transcriptions in Eudico, can also benefit from language processing tools and viewers from GATE.

The experiment lead to gaining an in-depth understanding of the functionality and operation of each of the two frameworks in isolation, and the benefits of their combined use. The showcase application exemplifies how the strengths of each framework have been combined to achieve seamless integration between speech, transcribed utterances, and linguistic data. The novel aspect of this integration is the resulting close inter-operation of the two infrastructures, which allows bi-directional data exchange and embedding of GUI components. Technically this is much more difficult to achieve than the usual case where one system uses the other through wrapper code. The application also proved the feasibility of the data- and GUI-reuse emphasis in GATE, as well as the extendibility of Eudico's dynamic viewers.

The inter-operation was made possible by the openness and flexibility of the underlying data models. Eudico's abstract corpus model showed its generality by the straightforward implementation of GATE/TIPSTER support. In turn, the generality of the GATE/TIPSTER model allowed efficient encoding and manipulation of transcribed speech data and the associated media time information.

From end-user perspective, the Eudico/GATE integrated application has introduced language engineering to the domain of spoken language research. In this way, linguists collecting and annotating speech/video corpora can also encode, manipulate and view linguistic data. From GATE programming perspective, the application highlighted the need for supporting different media and distributed processing. GATE version 2 which is currently under development, is aiming to address these issues.

References

H. Brugman, A. Russel, P. Wittenburg, and R. Piepenbrock. 1998. Corpus-based Research using the Internet. In *Workshop on Distributing and Accessing Linguistic Resources*, pages 8–15, Granada, Spain. `http://www.dcs.shef.ac.uk/~hamish/dalr/`.

H. Cunningham, K. Humphreys, R. Gaizauskas, and Y. Wilks. 1997. Software Infrastructure for Natural Language Processing. In *Proceedings of the Fifth Conference on Applied Natural Language Processing (ANLP-97)*, March. `http://xxx.lanl.gov/abs/cs.CL/9702005`.

H. Cunningham, R.G. Gaizauskas, K. Humphreys, and Y. Wilks. 1999. Experience with a Language Engineering Architecture: Three Years of GATE. In *Proceedings of the AISB'99 Workshop on Reference Architectures and Data Standards for NLP*, Edinburgh, April. The Society for the Study of Artificial Intelligence and Simulation of Behaviour.

R. Grishman and B. Sundheim. 1996. Message understanding conference - 6: A brief history. In *Proceedings of the 16th International Conference on Computational Linguistics*, Copenhagen, June.

B. MacWhinney. 1999. *The CHILDES Project: Tools for Analysing Talk (second ed.)*. Lawrence Arlbaum, Hillsdale, N.J.