DialogueView: an Annotation Tool for Dialogue

Fan Yang and **Peter A. Heeman**
Center for Spoken Langauge Understanding
OGI School of Science & Engineering
Oregon Health & Science University
20000 NW Walker Rd., Beaverton OR, U.S.A. 97006
{fly, heeman}@cslu.ogi.edu

1 Introduction

There is growing interest in collecting and annotating corpora of language use. Annotated corpora are useful for formulating and verifying theories of language interaction, and for building statistical models to allow a computer to naturally interact with people.

A lot of annotation tools have been built or are being built. CSLU Toolkit (Sutton et al., 1998) and Emu (Cassidy and Harrington, 2001) are built for words transcription or speech events (such as accent); DAT is built for coding dialogue acts using the DAMSL scheme (Core and Allen, 1997); Nb is built for annotating hierarchical discourse structure (Flammia, 1998); annotation toolkits, such as Mate (McKelvie et al., 2001), AGTK (Bird et al., 2001), and Nite (Carletta et al., 2003), are built for users to create their own tools. In this demo, we will present a novel tool, DialogueView, for annotating speech repairs, utterance boundaries, utterance tags, and hierarchical discourse structure altogether.

The annotation tool, DialogueView, consists of three views: WordView, UtteranceView, and BlockView. These three views present different abstractions of a dialogue, which helps users better understand what is happening in the dialogue. WordView shows the words time-aligned with the audio signal. UtteranceView shows the dialogue as a sequence of utterances. It abstracts away from the exact timing of the words and can even skip words, based on WordView annotations, that do not impact the progression of the dialogue. BlockView shows the dialogue as a hierarchy of discourse blocks, and abstracts away from the exact utterances that were said. Annotations are done at the view that is most appropriate for what is being annotated. The tool allows users to easily navigate among the three views and it automatically updates all views when changes are made in one view.

DialogueView makes use of multiple views to present different abstractions of a dialogue to users. Abstraction helps users focus on what is important for different annotation tasks. For example, for annotating speech repairs, utterance boundaries, and overlapping and abandoned utterances, WordView provides the exact timing information. For coding speech act tags and hierarchical discourse structure, UtteranceView shows a broader context and hides such low-level details.

In this presentation, we will show how DialogueView helps users annotate speech repairs, utterance boundaries, utterance tags, and hierarchical discourse blocks. Researchers studying dialogue might want to use this tool for annotating these aspects of their own dialogues. We will also show how the idea of abstraction in DialogueView helps users understand and annotate a dialogue. Although DialogueView focuses on spoken dialogue, we feel that abstraction can be used in annotating monologues, multi-party, and multi-modal interaction, with any type of annotations, such as syntactic structure, semantics and co-reference. Researchers might want to adopt the use of abstraction in their own annotation tools.

2 WordView

The first view is *WordView*, which takes as input two audio files (one for each speaker), the words said by each speaker and the start and stop times of each word (in XML format), and shows the words time-aligned with the audio signal. This view is ideal for seeing the exact timing of speech, especially overlapping speech. Users can annotate speech repairs, utterance boundaries, and utterance tags in WordView.

WordView gives users the ability to select a region of the dialogue and to play it. Users can play each speaker channel individually or both combined. Furthermore, DialogueView allows users to aurally verify their speech repair annotations. WordView supports playing a region of speech but with the annotated reparanda and editing terms skipped over. We have found this useful in deciding whether a speech repair is correctly annotated. If one has annotated the repair correctly, the edited speech will sound fairly natural.

3 UtteranceView

The annotations in WordView are utilized in building the next view, *UtteranceView*. This view shows the utterances of two speakers as if it were a script for a movie. To derive a single ordering of the utterances of the two speakers, we use the start time of each utterance as annotated in WordView. We refer to this process as *linearizing* the dialogue (Heeman and Allen, 1995). The order of the utterances should show how the speakers are sequentially adding to the dialogue, and is our motivation for defining utterances as being small enough so that they are not affected by subsequent speech of the other speaker.

Users can annotate utterance tags in UtteranceView besides WordView. WordView is more suitable for tags that depend on the exact timing of the words, or a very local context, such as whether an utterance is abandoned or incomplete, or whether there is overlap speech. UtteranceView is more suitable for tags that relate the utterance to other utterances in the dialogue, such as whether an utterance is an answer, a statement, a question, or an acknowledgment. Whether an annotation tag can be used in WordView or UtteranceView (or both) is specified in the configuration file. Which view a tag is used in does not affect how it is stored in the annotation files (also in XML format).

In UtteranceView, users can annotate hierarchical groupings of utterances. We call each grouping a *block*, and blocks can have other blocks embedded inside of them. Each block is associated with a *summary*, which users need to fill in. Blocks can be closed; when a block is closed, it is replaced by its summary, which is displayed as if it were said by the speaker who initiated the block. Just as utterances can be tagged, so can discourse blocks. The block tags scheme is also specified in the configuration file.

UtteranceView supports two types of playback. The first playback simply plays both channels mixed, which is exactly what is recorded. The second playback is slightly different. It takes the linearization into account and dynamically builds an audio file in which each utterance in turn is concatenated together, and a 0.5 second pause is inserted between each utterance. This gives the user an idealized rendition of the utterances, with overlapping speech separated. By comparing these two types of playbacks, users can aurally check if their linearization of the dialogue is correct.

Users can use the configuration file to customize UtteranceView. Typically, UtteranceView gives users a *clean display* of what is going on in a dialogue. This clean display removes reparanda and editing terms in speech repairs, and it also removes abandoned speech, which has no contributions to the conversation.[1] UtteranceView also supports adding texts or symbols to an utterance based on the tags, such as adding "?" after a question, "..." after an incomplete utterance, and "+" at both the beginning and end of an overlapping utterance to signal the overlap. (c.f. *Childes* scheme (MacWhinney, 2000)).

[1]Note that these clean processes are optional. Users can specify them in the configuration file.

4 BlockView

In addition to WordView and UtteranceView, we are experimenting with a third view, which we call *BlockView*. This view shows the hierarchical structure of the discourse by displaying the summary and intention (DSP) for each block, indented appropriately. BlockView gives a very concise view of the dialogue. It is also convenient for navigating in the dialogue. By highlighting a line and then pressing *Sync*, the user can see the corresponding part of the dialogue in UtteranceView and WordView.

5 Availability

DialogueView is written in Incr Tcl/Tk. We also use the snack package for audio support; hence DialogueView supports audio file formats of WAV, MP3, AU, and others (see *http://www.speech.kth.se/snack/* for the complete list). DialogueView has been tested on Microsoft Windows (2000 and XP) and Redhat Enterprise Linux.

DialogueView is freely available for research and educational use. Users should first install a standard distribution of Tcl/Tk, such as ActiveTcl from *http://www.tcl.tk*, and then download DialogueView from *http://www.cslu.ogi.edu/DialogueView*. The distribution also includes some examples of annotated dialogues.

References

Steven Bird et al. 2001. Annotation tools based on the annotation graph API. In *Proceedings of ACL/EACL 2001 Workshop on Sharing Tools and Resources for Research and Education*.

Jean Carletta et al. 2003. The NITE XML toolkit: flexible annotation for multi-modal language data. *Behavior Research Methods, Instruments, and Computers*, April. Special Issue on Measuring Behavior.

Steve Cassidy and Jonathan Harrington. 2001. Multi-level annotation in the Emu speech database management system. *Speech Communication*, 33:61–77.

Mark G. Core and James F. Allen. 1997. Coding dialogues with the DAMSL annotation scheme. In *Proceedings of AAAI Fall 1997 Symposium*.

Giovanni Flammia. 1998. *Discourse Segmentation Of Spoken Dialogue: An Empirical Approach*. Ph.D. thesis, Massachusetts Institute of Technology.

Peter A. Heeman and James Allen. 1995. Dialogue transcription tools. Trains Technical Note 94-1, URCS, March.

Brian MacWhinney. 2000. *The CHILDES Project: Tools for Analyzing Talk*. Mahwah, NJ:Lawrence Erlbaum Associates, third edition.

D. McKelvie, et al. 2001. The MATE Workbench - An annotation tool for XML coded speech corpora. *Speech Communication*, 33(1-2):97–112. Special issue, "speech Annotation and Corpus Tools".

Stephen Sutton et al.. 1998. Universal speech tools: The CSLU toolkit. In *Proceedings of 5th ICSLP*, Australia.