DialogueView: An Annotation Tool for Dialogue

Peter A. Heeman and **Fan Yang** and **Susan E. Strayer**
Computer Science and Engineering
OGI School of Science and Engineering
Oregon Health & Science University
20000 NW Walker Rd., Beaverton OR, 97006
heeman@cse.ogi.edu yangf@cse.ogi.edu susan_strayer@yahoo.com

Abstract

This paper describes DialogueView, a tool for annotating dialogues with utterance boundaries, speech repairs, speech act tags, and discourse segments. The tool provides several views of the data, including a word view that is time-aligned with the audio signal, and an utterance view that shows the dialogue as if it were a script for a play. The utterance view abstracts away from lower level details that are coded in the word view. This allows the annotator to have a simpler view of the dialogue when coding speech act tags and discourse structure, but still have access to the details when needed.

1 Introduction

There is a growing interest in annotating human-human dialogue. Annotated dialogues are useful for formulating and verifying theories of dialogue and for building statistical models. In addition to orthographic word transcription, one might want the following dialogue annotations.

- Annotation of the speech repairs. Speech repairs are a type of disfluency where speakers go back and change or repeat something they just said.
- Segmentation of the speech of each speaker into utterance units, with a single ordering of the utterances. We refer to this as *linearizing* the dialogue (Heeman and Allen, 1995a).
- Each utterance tagged with its speech act function.
- The utterances grouped into hierarchical discourse segments.

There are tools that address subsets of the above tasks. However, we feel that doing dialogue annotation is very difficult. Part of this difficulty is due to the interactions between the annotation tasks. An error at a lower level can have a large impact on the higher level annotations. For instance, there can be ambiguity as to whether an "okay" is part of a speech repair; this will impact the segmentation of the speech into utterance units and the speech act coding. Sometimes, it is only by considering the higher level annotations that one can make sense of what is going on at the lower levels, especially when there is overlapping speech. Hence, a tool is needed that lets users examine and code the dialogue at all levels. The second reason why dialogue annotation is difficult is because it is difficult to follow what is occurring in the dialogue, especially for coding discourse structure. A dialogue annotation tool needs to help the user deal with this overload.

In this paper, we describe our dialogue annotation tool, *DialogueView*. This tool displays the dialogue at three different levels of abstraction. The word level shows the words time-aligned with the audio signal. The utterance level shows the dialogue as a sequence of utterance, as if it were a script for a play. It abstracts away from the exact timing of the words and even skips words that do not impact the progression of the dialogue. The block level shows the dialogue as a hierarchy of discourse segment purposes, 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 the user to easily navigate between the three views and automatically updates the higher level views when changes are made in the lower level views. Because the higher levels abstract away lower level details, it is easier for the user to understand what is happening in the dialogue.[1] Yet, the user can easily refer to the lower level views to see what

[1] This approach bears resemblance to the work of Jönsson and Dahlbäck (2000) in which they *distill* human-human dialogues by removing those parts that would not occur in human-computer dialogue. They do this to create training data for their spoken dialogue systems.

actually occurred when necessary.

In the rest of this paper, we first discuss other annotation tools. We then describe the three levels of abstraction in our annotation tool. We then discuss audio playback. Next, we discuss how the tool can be customized for different annotation schemes. Finally, we discuss the implementation of the tool.

2 Existing Tools

There are a number of tools that let users annotate speech audio files. These include *Emu* (Cassidy and Harrington, 2001), *SpeechView* (Sutton et al., 1998) and *Waves+* (Ent, 1993). These tools often allow multiple text annotation files to be associated with the waveform and allow users an easy facility to capture time alignments. For instance, for the ToBI annotation scheme (Pitrelli et al., 1994), one can have the word alignment in one text file, intonation features in a second, break indices in a third, and miscellaneous features in a fourth. The audio annotation tools often have powerful signal analysis packages for displaying such phenomena as spectrograms and voicing. These tools, however, do not have any built-in facility to group words into utterances nor group utterances into hierarchical discourse segments.

Several tools allow users to annotate higher level structure in dialogue. The annotation tool *DAT* from the University of Rochester (Ferguson, 1998) allows users to annotate utterances or groups of utterances with a set of hard-coded annotation tags for the DAMSL annotation scheme (Allen and Core, 1997; Core and Allen, 1997). The tool *Nb* from MIT (Flammia, 1995; Flammia, 1998) allows users to impose a hierarchical grouping on a sequence of utterances, and hence annotate discourse structure. Both DAT and Nb take as input a linearization of the speaker utterances. Mistakes in this input cannot be fixed. Whether there are errors or not, users cannot see the exact timing interactions between the speakers' words or the length of pauses. This simplification can make it difficult for the annotator to truly understand what is happening in the dialogue, especially for overlapping speech, where speakers fight over the turn or make back-channels.

The tools *Transcriber* (Barras et al., 2001) and *Mate* (McKelvie et al., 2001) allow multiple views of the data: a word view with words time-aligned to the audio signal and an utterance view. However, *Transcriber* is geared to single channel data and has a weak ability to handle overlapping speech and *Mate* only allows one view to be shown at a time. The author of a competing tool has remarked that "speed and stability of [Mate] are both still problematic for real annotation. Also, the highly generic approach increases the initial effort to set up the tool since you basically have to write your own tool using the Mate style-sheet language" (Kipp, 2001). Hence, he developed a new tool *Anvil* that he claims is a better trade-off among generality, functionality, and complexity. This tool offers multi-modal annotation support, but like Transcriber, does not allow detailed annotation of dialogue phenomena, such as overlapping speech and abandoned speech, and has no abstraction mechanism.

3 Word View

Our dialogue annotation tool, DialogueView, gives the user three views of the data. The lowest level view is called *WordView* and takes as input the words said by each speaker and their start and stop times and shows them time-aligned with the audio signal. Figure 1 shows an excerpt from the Trains corpus (Heeman and Allen, 1995b) in WordView. This view is ideal for viewing the exact timing of speech, especially overlapping speech. As will be discussed below, we use it for segmenting the speech into utterances, and annotating communicative status and speech repairs.[2] These annotations will allow us to build a simpler representation of what is happening in the speech for the utterance view, which is discussed in the next section.

3.1 Utterance Segmentation

As can be seen in Figure 1, WordView shows the words segmented into *utterances*, which for our purpose is simply a grouping of consecutive words by a single speaker, in which there is minimal effect from the other speaker. Consider the exchange in Figure 1, where the upper speaker says "and then take the remaining boxcar down to" followed by the lower speaker saying "right, to du-", followed by the upper speaker saying "Corning". Although one could consider the upper speaker's speech as one utterance, we preclude that due to the interaction from the lower speaker. A full definition of 'utterance' is beyond the scope of this paper, and is left to the users to specify in their annotation scheme.

Utterance boundaries in WordView are only shown by their start times. The end of the utterance is either the start of the next utterance by the

[2]There currently is no support for changing the word transcription. There are already a number of tools that do an excellent job at this task. Hence, adding this capability is a low priority for us.

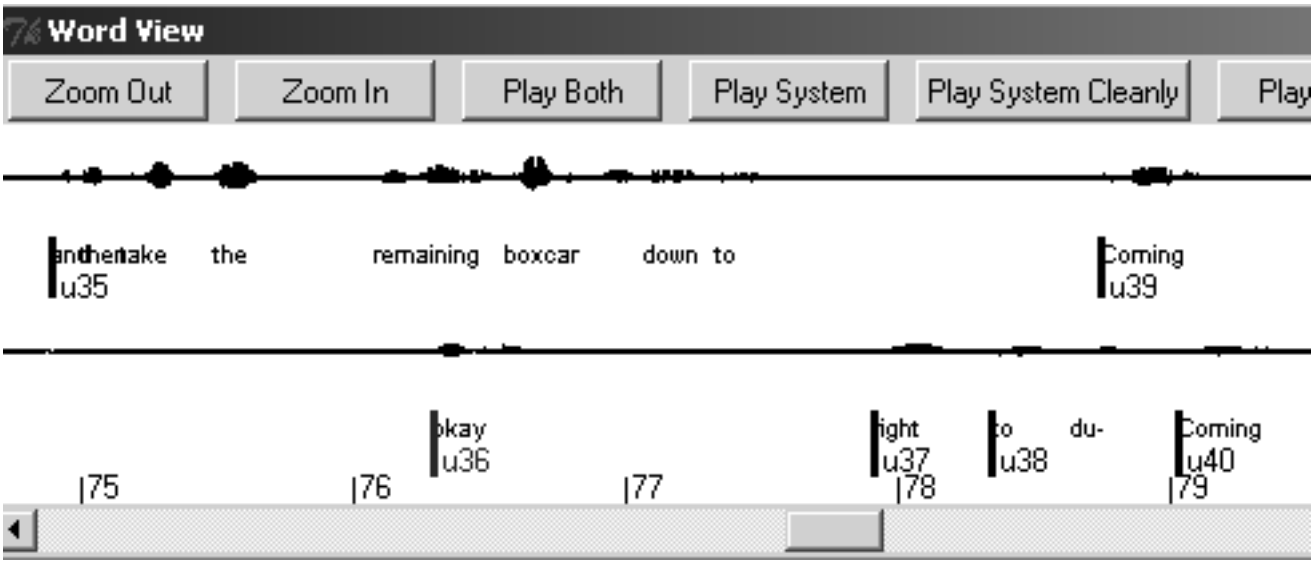

Figure 1: Utterance Segmentation in WordView

same speaker or the end of the file. With WordView, the user can easily move, insert or delete utterance boundaries. The tool ensures that the boundaries never fall in the middle of a word by the speaker.

The start times of the utterances are used to derive a single ordering of the utterances of the two speakers. This linearization of the dialogue captures how the speakers are sequentially adding to the dialogue. The linearization is used by the next view to give an abstraction away from the exact timing of the speech.

3.2 Communicative Status

WordView allows features of the utterances to be annotated. In practice, however, we only annotate features related to its *communicative status*, based on the DAMSL annotation scheme (Allen and Core, 1997). Below, we give the utterance tags that we assign in WordView.

Abandoned: The speaker abandoned what they were saying and it had no impact on the rest of the dialogue. This mainly happens where one speaker loses the fight over who speaks next. Figure 2 gives an example where the upper speaker tries to take the turn by saying "takes," and than abandons this effort.

Incomplete: The speaker did not make a complete utterance, either prosodically, syntactically, or semantically. Unlike the previous category, the partial utterance did impact the dialogue behavior. Figure 1 gives two examples. The first is where the upper speaker says "and than take the remaining boxcar down to", and the second is where the lower speaker said "to du-," which was than completed by the upper speaker.

Overlap: The speech in the utterance overlaps with the speech from the prior utterance. For instance, in Figure 1, the lower speaker utters "okay" during the middle of an utterance, perhaps to tell the upper speaker that they are understanding everything so far. However, a simple linearization would make it seem that the "okay" is an acknowledgment of the entire utterance, which is not the case. Hence, we tag the "okay" utterance with the overlap tag. The next view, UtteranceView, will take the overlap tag into account in displaying the utterances, as will be discussed in Section 4.3.

Not all overlapping speech needs to be annotated with the overlap tag. In Figure 1, the second instance of "Corning" overlaps slightly with the first instance of "Corning". However, viewing it sequentially does not alter the analysis of the exchange.

3.3 Reordering Overlapping Speech

Consider the example in Figure 2. Before the excerpt shown, the lower speaker had just finished an utterance and then paused for over a second. The upper speaker then acknowledged the utterance with "okay", but this happened a fraction of a second after the bottom speaker started speaking again. A simple linearization of the dialogue would have the "okay" following the wrong stretch of speech—"and than th(at)- takes w- what three hours." A solution to this would be to anno-

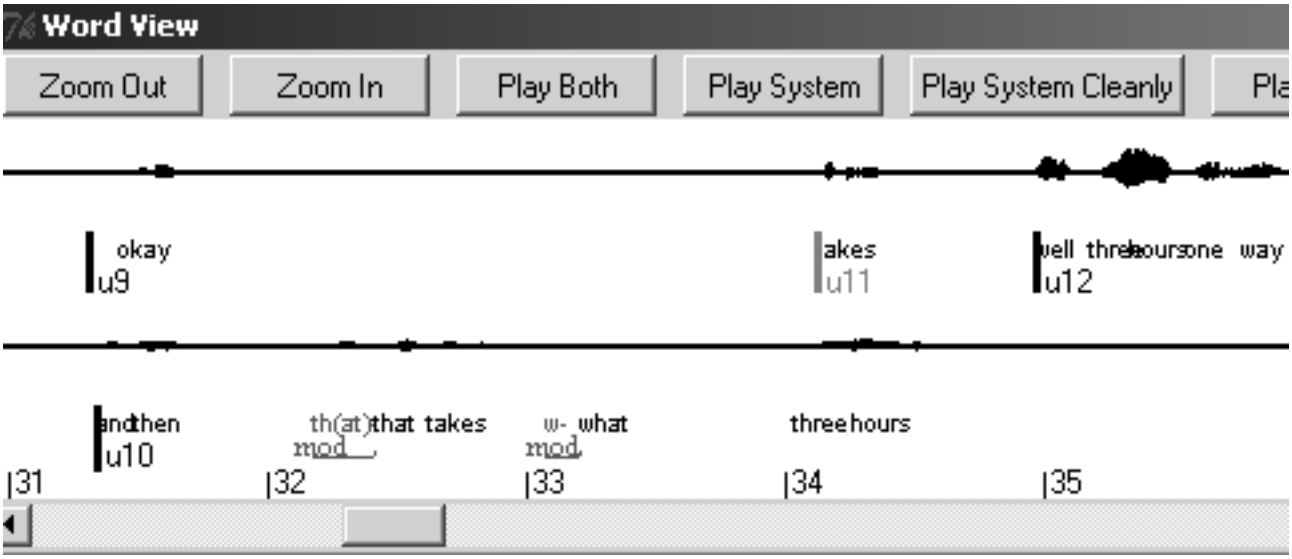

Figure 2: Utterance Ordering and Speech Repairs in WordView

tate the "okay" with the overlap tag. However, this "Okay" is more similar to the overlapping instances of "Corning" in Figure 1. The fact that "Okay" overlaps with the start of the next utterance is not critical to understanding what is occurring in the dialogue, as long as we linearize the "Okay" to occur before the other utterance. WordView allows the user to change the linearization by moving the start times of utterances. This can be done provided that the speaker was silent in the time interval preceding where the other person started talking.

In summary, overlapping speech can be handled in three ways. The utterance can be explicitly tagged as overlapping; the overlap can be ignored if it is not critical in understanding what is going on in the dialogue; or the start time of the utterance can be changed so that the overlap does not need to be tagged.

3.4 Speech Repairs

WordView also allows users to annotate speech repairs. A speech repair is where a user goes back and repeats or changes something that was just said (Heeman and Allen, 1999). Below we give an example of a speech repair and show its principle components: *reparandum*, *interruption point*, and *editing term*.

***Example* 1**
why don't we $\underbrace{\text{take}}_{\textit{reparandum}}$ $\uparrow_{\textit{ip}}$ $\underbrace{\text{um}}_{\textit{et}}$ take two boxcars

The reparandum is the speech that is being replaced, the interruption point is the end of the reparandum, and the editing term consists of words such as "um", "uh", "okay", "let's see" that help signal the repair.

To annotate a repair, the user highlights a sequence of words and then tags it as a reparandum or an editing term of a repair. The user can also specify the type of repair. Figure 2 shows how speech repairs are displayed in WordView. The words in the reparandum and editing term are underlined and displayed in a special color.

3.5 Speech Repairs and Utterances

Some phenomena can be marked as either a speech repair, or could be marked using the utterance tags of *incomplete* or *abandon*. This is especially true for fresh starts (Heeman and Allen, 1999), where a speaker abandons the current utterance and starts over. To avoid having multiple ways of annotating the same phenomena, we impose the following restrictions in our annotation scheme.

- There cannot be an utterance boundary inside of a reparandum, inside of an editing term, at the interruption point, nor at the end of the editing term. Hence, something annotated as a reparandum cannot also be annotated as an abandoned utterance.

- Abandoned or incomplete utterances cannot be followed by an utterance by the same speaker.

- All word fragments must either be the last word of a reparandum or the last word of an utterance that is marked as abandoned or incomplete.

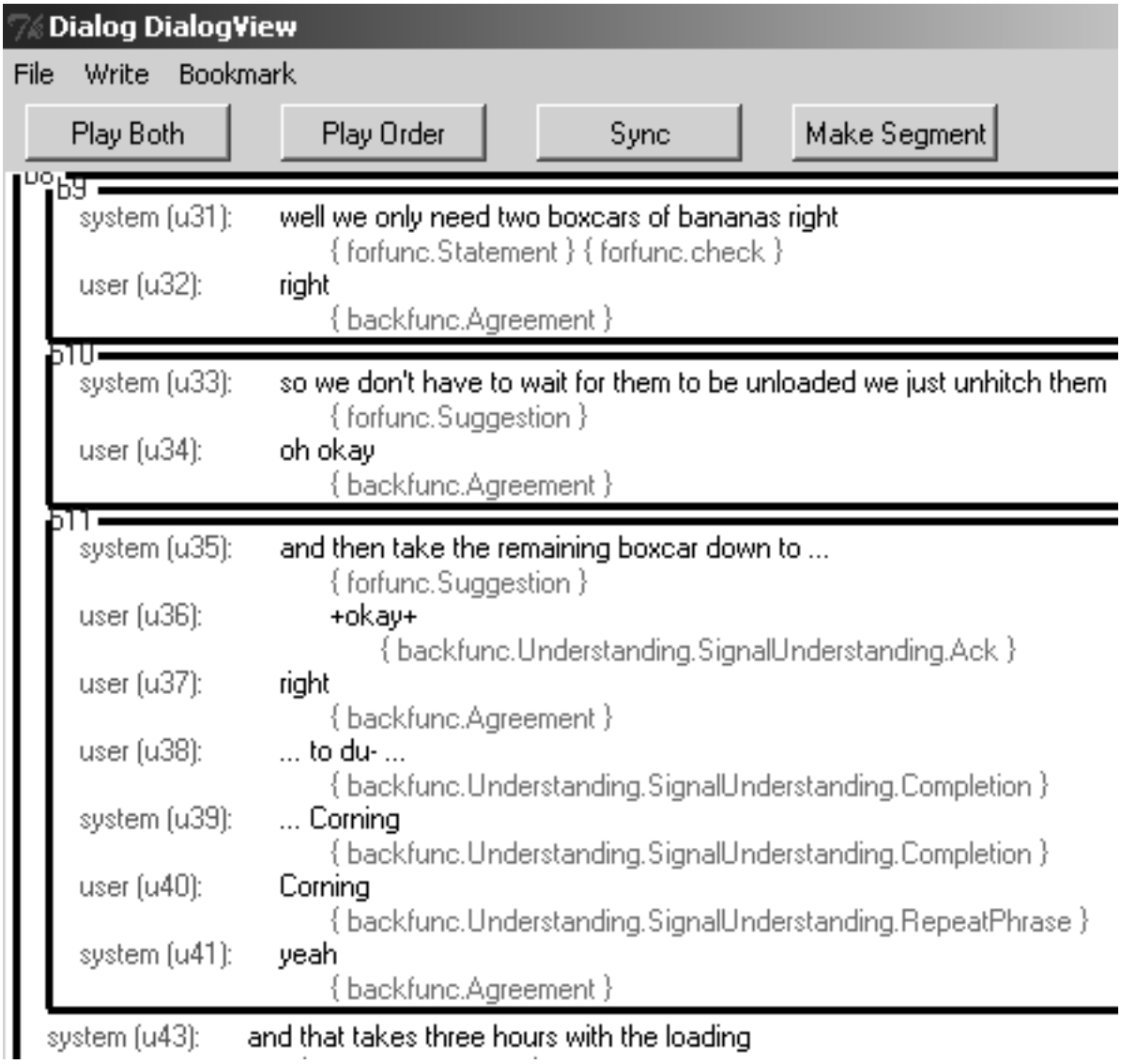

Figure 3: UtteranceView: Segmented Utterances in UtteranceView

- Abandoned or incomplete utterances can end with an editing term, which would be marked as the editing term of an abridged repair.

3.6 Summary

There are a number of reasons why we annotate utterance boundaries, speech repairs, and communicative status in WordView. Annotating utterance boundaries and overlapping speech requires the user to take into account the exact timing of the utterances, which is best done in this view. Speech repairs also require fine tuned listening to the speech and have strong interactions with utterance boundaries. Furthermore, all three types of annotations can be used to build a simpler view of what is happening in the dialogue, as will be explained in the next section.

4 Utterance View

The annotations from the word view are used to build the next view, which we refer to as *UtteranceView*. The dialogue excerpts from Figures 1 and 2 are shown in the utterance view in Figures 3 and 4, respectively. The utterance view abstracts away from the detailed timing information and individual words that were spoken. Instead, it focuses on the sequence of utterances between the speakers. By removing details that were annotated in the word view, we still preserve the important details that are needed to annotate speech act types for the utterances and to annotate discourse segments. Of course, if the user wants to see the exact timing of the words in the utterances, they can examine the word view, as it is

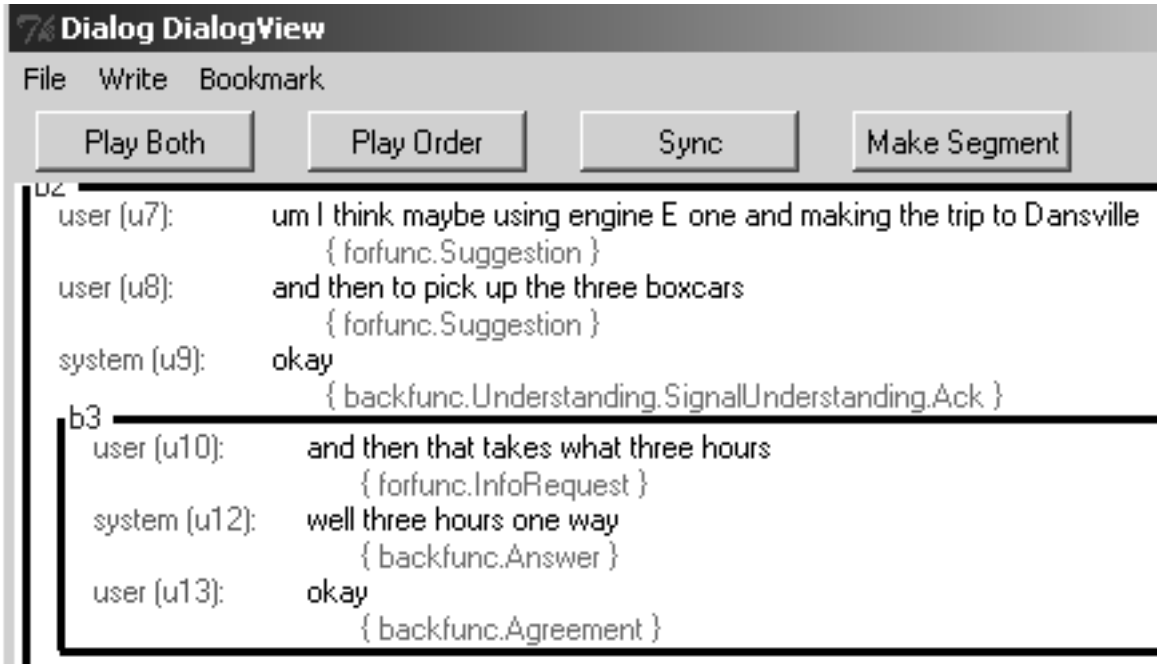

Figure 4: UtteranceView: Reordered and Abandoned Utterances in UtteranceView

displayed alongside the utterance view. There are also navigation buttons on each view that allow the user to easily reposition the portion of the dialogue in the other view. Furthermore, changes made in the word view are immediately propagated into the utterance view, and hence the user will immediately see the impact of their annotations.

4.1 Utterance Ordering

Utterance ordering in the utterance view is determined by the start times of the utterances as specified in WordView. As was explained earlier, altering the start time of an utterance can be used to simplify some cases of overlapping speech, where the overlap is not critical to understanding the role of the utterance in the dialogue. Figure 2 gave the word view of such an example. Rather than code it as an overlap, we moved the start time of the "okay" utterance so that it precedes the overlapping speech by the other speaker. Figure 4 shows how this looks in the utterance view. Here, the annotator would view the "okay" as an acknowledgment that occurred between the two utterances of the lower speaker.

4.2 Speech Repairs

In the word view, the user annotates the reparandum and editing term of speech repairs. If the reparandum and editing term are removed, the resulting utterance reflects what the speaker intended to say. Speech repairs do carry information.

- Their occurrence can signal a lack of certainty of the speaker.
- The reparandum of a repair can have an anaphoric reference, as in "Peter was, well he was fired."

However, removing the reparandum and editing term of speech repairs from utterances in the utterance view leads to a simpler representation of what is occurring in the dialogue. Hence, in the utterance view, we clean up the speech repairs, as shown in Figures 2 and 4. Figure 2, which shows the word view, contains the utterance "and then th(at) that takes w- what three hours"; whereas Figure 4, which shows the utterance view, contains "and then that takes what three hours." Of course, a user can always refer to the word view when annotating in the utterance view if they want to see the exact speech that was said. In most cases, we feel that this will not be necessary for annotating speech act tags and discourse segments.

4.3 Communicative Status

The communicative status coded in the word view is used in formatting the utterance view. Utterances tagged as overlapping are indented and displayed with '+' on either side, as shown in Figure 3. Utterances tagged as abandoned are not

shown, as can be seen in Figure 4, in which the abandoned utterance "takes" made by the upper speaker is not included. Utterances tagged as incomplete are shown with a trailing "..." as shown in Figure 3.

4.4 Annotating Utterance Tags

In the utterance view, one can also annotate the utterances with various tags. For our work, we use a subset of the DAMSL tags corresponding to forward and backward functions (Allen and Core, 1997). Forward functions include statement, request information, and suggestion. Backward functions include answer, acknowledgment, and agreement. Although these utterance tags could be annotated in the word view, doing it in the utterance view allows us to see more context, which is needed to give the utterance the proper tags. When necessary, the annotator can easily refer to the word view to see the exact local context.

4.5 Annotating Blocks of Utterances

In the utterance view, the user can also annotate hierarchical groupings of utterances.[3] We use the utterance blocks to annotate discourse structure (Grosz and Sidner, 1986). This is similar to what Flammia's tool allows (Flammia, 1995). Rather than showing it with indentation and color, we draw boxes around segments. Figure 3 shows a dialogue excerpt with three utterance blocks inside of a larger block. To create a segment, the user highlights a sequence of utterances and then presses the "make segment" button. The user can change the boundaries of the blocks by simply dragging either the top or bottom edge of the box. Blocks can also be deleted. The tool ensures that if two blocks have utterances in common then one block is entirely contained in the other.

Tags can also be assigned to the blocks. We have just started using the tool for discourse segmentation, and so we are still refining these tags. In Grosz and Sidner's theory of discourse structure (1986), the speaker who initiates the block does so to accomplish a certain purpose. We have a tag for annotating this purpose. We also have tags to categorize the block as a greeting, specify goal, construct plan, summarize plan, verify plan, give info, request info, or other (Strayer and Heeman, 2001).

The utterance view also allows the user to open or close a block. When a block is open (the default), all of its utterances and sub-blocks are displayed. When it is closed, its utterances and sub-blocks are replaced by the single line purpose. Opening and closing blocks is useful as it allows the user to control the level of detail that is shown. Consider the third embedded block shown in Figure 3, in which the conversants take seven utterances to jointly make a suggestion. After we have analyzed it, we can close it and just see the purpose. This will make it easier to determine the segmentation of the blocks that contain it.

We are experimenting with a special type of dialogue block. Consider the example from the previous paragraph, in which the conversants take seven utterances to jointly make a suggestion. This is related to the *shared turns* of Schiffrin (1987), the *co-operative completions* of Linell (1998), and the grounding units of Traum and Nakatani (1999). We are experimenting with how to support the annotation of such phenomena. We have added a tag to indicate whether the utterances in the block are being used to build a single contribution. For these single contributions, we also supply a concise paraphrase of what was said. We have found that this paraphrase can be built from a sequential subset of the words in the utterances of the block. For instance, the paraphrase of our example block is "and then take the remaining boxcar down to Corning."

5 Block View

We are experimenting with a third view of the dialogue. This view, which we refer to as *Block-View*, abstracts away from the individual utterances, and shows the hierarchical structure of the discourse segments. This gives a very concise view of the dialogue. The block view is also convenient for it provides an index to the whole dialogue. This allows the user to quickly move around the dialogue.

6 Audio Playback

Each view gives the user the ability to select a region of the dialogue and to play it. In the word view, the user can play each speaker channel individually or both combined.[4] This ability is especially useful for overlapping speech, where the annotator would want to listen to what each speaker said individually, as well as hear the timing between the speaker utterances.

Just as each view provides a visual abstraction from the previous one, we also do the same with audio playback. In the word view, which has

[3] We do not allow segments to be interleaved. It is unclear if such phenomena actually occur.

[4] In order to play each speaker individually, we require a separate audio channel for each speaker.

```
wordViewUtt            => atmostoneof abandoned incomplete overlap
uttViewUtt             => anyof forward backward comment
uttViewUtt.forward     => oneof statement question suggestion other
uttViewUtt.backward    => oneof agreement understanding answer other
uttViewUtt.comment     => other
```

Figure 5: Sample Specification of Utterance Tags

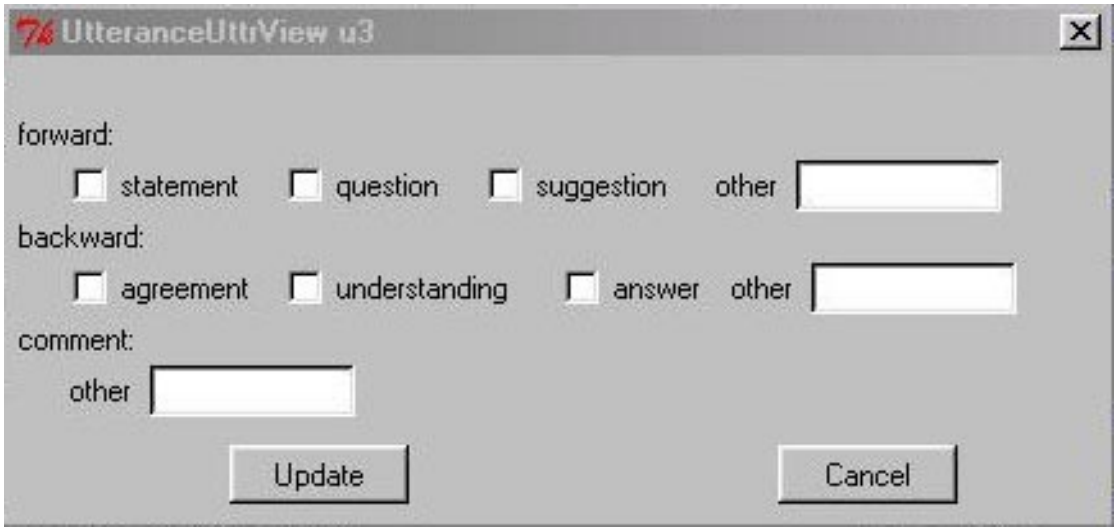

Figure 6: Sample Utterance Annotation Panel

the speech repairs annotated, the user can play back the speech *cleanly* of either speaker, where the stretches of speech annotated as the reparandum or editing term of a repair are skipped. We have found this to be of great assistance in verifying if something should be annotated as a repair or not. It gives us an audio means to verify the speech repair annotations. If we have annotated the repair correctly, the edited audio signal should sound fairly natural.

In formatting the utterance view, we take into account whether utterances have been marked as abandoned or overlapped. We provide a special playback in the utterance view that takes this into account. We build an audio file in which we skip over repairs, skip over abandoned speech, and shorten large silences. If there is overlap that is not marked as significant, we linearize it by concatenating the utterances together. If the overlap is marked as significant, we keep the overlap. We are finding that this gives us an audio means to ensure that our markings of abandonment, overlap and our linearization is correct.

We are also experimenting with even further simplifying the audio output. For blocks that have a paraphrase, and the block is closed, we play the paraphrase by constructing it from the words said in the block. For blocks that are closed that do not have a paraphrase, we use the text-to-speech engine in the CSLU toolkit (Colton et al., 1996; Sutton et al., 1997) to say the purpose, as if there was a narrator.

7 Customizations

Some aspects of the tool are built in, such as the notion of utterances, speech repairs, and hierarchical grouping of utterances into blocks. However, the annotations of these phenomena and how they are displayed can be customized through a configuration file. This allows us to easily experiment as we revise our annotation scheme; to use domain specific tags; and to make the tool useful for other researchers who might use different tags.

Speech repair tags, utterance tags, and block tags are specified in the configuration file. Figure 5 gives a sample of how the annotation tags for an utterance are specified. The two top level entries in the figure are "wordViewUtt" and "uttViewUtt", which specify the utterance annotation tags in WordView and UtteranceView, respectively. The decomposition can be of one of three types.

atmostoneof: at most one of the attributes can

be specified

oneof: exactly one of the attributes must be specified

anyof: there is no restriction on which attributes can be specified

The subcomponents can either be terminals as is the case for the decomposition of "wordViewUtt", or can be non-terminals, as is the case for each of the three subcomponents of "uttViewUtt". Hence, hierarchical tags are supported. Terminals are assumed to be of binary type, except for "other", which is assumed to be a string. The configuration file determines how the annotation panel is generated. For the annotation scheme specified in Figure 5, Figure 6 shows the annotation panel that would be automatically generated for the utterance view.

As we explained earlier, some of the utterance tags affect how the word view and utterance view are formatted. Rather than hard code this functionality, it is specified in the configuration file. We are still experimenting with the best way to code this functionality. Figure 7 gives an example of how we code the utterance tag functionality. The first line indicates that the utterance

```
wordViewUtt.abandoned  do wordView color red
wordViewUtt.abandoned  uttView  ignore
wordViewUtt.incomplete wordView color yellow
wordViewUtt.incomplete uttView  trailsoff
wordViewUtt.overlap    wordView color blue
wordViewUtt.overlap    uttView  overlap
```

Figure 7: Sample Utterance Display Specification

tag of "abandoned" coded in WordView should be displayed in red in WordView. The second line indicates that it should not be displayed in UtteranceView.

8 Implementation

DialogueView is written in Tcl/Tk. We also use utilities from the CSLU Speech Toolkit (Colton et al., 1996; Sutton et al., 1997), including audio and wave handling and speech synthesis. We have rewritten the tool to use an object-oriented extension of Tcl called Tclpp, designed and developed by Stefan Simnige. This is allowing us to better manage the growing complexity of the tool as well as reuse pieces of the software in our annotation comparison tool (Yang et al., 2002). It should also help in expanding the tool so that it can handle any number of speakers.

9 Conclusion

In this paper, we described a dialogue annotation tool that we are developing for segmenting dialogue into utterances, annotating speech repairs, tagging speech acts, and segmenting dialogue into hierarchical discourse segments. The tool presents the dialogue at different levels of abstraction allowing the user to both see in detail what is going on and see the higher level structure that is being built. The higher levels not only abstract away from the exact timing, but also can skip over words, whole utterances, and even simplify segments to a single line paraphrase. Along with the visual presentation, the audio can also be played at these different levels of abstraction. We feel that these capabilities should help annotators better code dialogue.

This tool is still under active development. In particular, we are currently refining how blocks are displayed, improving the ability to customize the tool for different tagsets, and improving the audio playback facilities. As we develop this tool, we are also doing dialogue annotation, and refining our scheme for annotating dialogue in order to better capture the salient features of dialogue and improve the inter-coder reliability.

10 Acknowledgments

The authors acknowledgment funding from the Intel Research Council.

References

James F. Allen and Mark G. Core. 1997. Damsl: Dialog annotation markup in several layers. Unpublished Manuscript.

Claude Barras, Edouard Geoffrois, Zhibiao Wu, and Mark Liberman. 2001. Transcriber: development and use of a tool for assisting speech corpora production. *Speech Communications*, 33:5–22.

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

Don Colton, Ron Cole, David G. Novick, and Stephen Sutton. 1996. A laboratory course for designing and testing spoken dialogue systems. In *Proceedings of the International Conference on Audio, Speech and Signal Processing (ICASSP)*, pages 1129–1132.

Mark G. Core and James F. Allen. 1997. Coding dialogs with the DAMSL annotation scheme.

In *Working notes of the AAAI Fall Symposium on Communicative Action in Humans and Machines*.

Entropic Research Laboratory, Inc., 1993. *WAVES+ Reference Manual*. Version 5.0.

George Ferguson. 1998. DAT: Dialogue annotation tool. Available from www.cs.rochester.edu in the subdirectory research/speech/damsl.

Giovanni Flammia. 1995. N.b.: A graphical user interface for annotating spoken dialogue. In *AAAI Spring Symposium on Empirical Methods in Discourse Interpretation and Generation*, pages pages 40–46, Stanford, CA.

Giovanni Flammia. 1998. Discourse segmentation of spoken dialogue: an empirical approach. Doctoral dissertation, Department of Electrical and Computer Science, Massachusetts Institute of Technology.

Barbara J. Grosz and Candace L. Sidner. 1986. Attention, intentions, and the structure of discourse. *Computational Linguistics*, 12(3):175–204.

Peter A. Heeman and James Allen. 1995a. Dialogue transcription tools. Trains Technical Note 94-1, Department of Computer Science, University of Rochester, March. Revised.

Peter A. Heeman and James F. Allen. 1995b. The Trains spoken dialog corpus. CD-ROM, Linguistics Data Consortium, April.

Peter A. Heeman and James F. Allen. 1999. Speech repairs, intonational phrases and discourse markers: Modeling speakers' utterances in spoken dialog. *Computational Linguistics*, 25(4):527–572.

Arne Jönsson and Nils Dahlbäck. 2000. Distilling dialogues — a method using natural dialogue corpora for dialogue systems development. In *Proceedings of the 6th Applied Natural Language Processing Conference*, pages 44–51, Seattle.

Michael Kipp. 2001. Anvil: A generic annotation tool for multimodal dialogue. In *Proceedings of the 7th European Conference on Speech Communication and Technology (Eurospeech)*.

Per Linell. 1998. *Approaching Dialogue: Talk, Interaction and Contexts in Dialogical Perspectives*. John Benjamins Publishing.

David McKelvie, Amy Isard, Andreas Mengel, Morten Baun Mùller, Michael Grosse, and Marion Klein. 2001. The MATE workbench — an annotation tool for XML coded speech corpora. *Speech Communications*, 33:97–112.

John F. Pitrelli, Mary E. Beckman, and Julia Hirschberg. 1994. Evaluation of prosodic transcription labeling reliability in the ToBI framework. In *Proceedings of the 3rd International Conference on Spoken Language Processing (ICSLP-94)*, Yokohama, September.

Deborah Schiffrin. 1987. *Discourse Markers*. Cambridge University Press, New York.

Susan E. Strayer and Peter A. Heeman. 2001. Dialogue structure and mixed initiative. In *Second workshop of the Special Interest Group on Dialogue*, pages 153–161, Aalborg Denmark, September.

Stephen Sutton, Ed Kaiser, Andrew Cronk, and Ronald Cole. 1997. Bringing spoken language systems to the classroom. In *Proceedings of the 5th European Conference on Speech Communication and Technology (Eurospeech)*, Rhodes, Greece.

S. Sutton, R. Cole, J. de Villiers, J. Schalkwyk, P. Vermeulen, M. Macon, Y. Yan, E. Kaiser, R. Rundle, K. Shobaki, P. Hosom, A. Kain, J. Wouters, M. Massaro, and M. Cohen. 1998. Universal speech tools: the cslu toolkit. In *Proceedings of the 5th International Conference on Spoken Language Processing (ICSLP-98)*, pages 3221–3224, Sydney Australia, November.

David R. Traum and Christine H. Nakatani. 1999. A two-level approach to coding dialogue for discourse structure: Activities of the 1998 working group on higher-level structures. In *Proceedings of the ACL'99 Workshop Towards Standards and Tools for Discourse Tagging*, pages 101–108, June.

Fan Yang, Susan E. Strayer, and Peter A. Heeman. 2002. ACT: a graphical dialogue annotation comparison tool. Submitted for publication.