Dialogue Transcription Tools

Peter A. Heeman and James Allen

The University of Rochester
Computer Science Department
Rochester, New York 14627

TRAINS Technical Note 94-1

March 1995 Revised

Abstract

This document describes a toolkit and guidelines for the transcription of dialogues. The premise of these tools is that a dialogue between two people can be broken down into a series of utterance files, each spoken by one participant. This allows the transcription tools and standards already designed for single speaker speech to be used.

Funding gratefully received from the Natural Sciences and Engineering Research Council of Canada, from NSF under Grant IRI-90-13160, and from ONR/DARPA under Grant N00014-92-J-1512.

Contents

1 Introduction

In recent years, there has been an increasing amount of interest in spoken language. With this interest has come corpora of speech, tools to work with it, and a standard for transcribing it. For instance, there is the ATIS corpus of questions that have been asked to an automated reservation system (MADCOW, 1992); there is the WAVES system (Ent, 1993) for annotating audio files with time-aligned annotations; and there is the ToBI system (Silverman et al., 1992; Beckman and Hirschberg, 1994; Beckman and Ayers, 1994), a standard for transcribing single speaker utterances with intonational features and break indices. However, these resources are oriented towards single speaker utterances. Spoken dialogues, especially naturally occurring ones spoken by *untrained* speakers, exhibit a number of phenomena that simply do not occur in single speaker utterances, such as overlapping speech, back-channel responses, and turn-taking, and so need to be studied if we are to understand what is going on. In order to take advantage of the work that has gone into single speaker utterances, we need tools and guidelines for bridging the gap between dialogues and single speaker utterances.

At the University of Rochester, we are involved in a long term research project, the TRAINS project, to develop an intelligent planning assistant that is conversationally proficient in natural language (Allen and Schubert, 1991). In order to provide empirical data to drive our research, we are collecting dialogues between two people, one playing the role of the system and the other playing the role of a manager, who has a certain planning problem to solve. The first transcription effort, which did not make use of WAVES or the ToBI standard, is presented in Gross, Allen, and Traum (1993). In our current effort, we are transcribing the dialogues using WAVES and the ToBI standard. In order to do so, we have developed a standard and a toolkit for breaking a dialogue into individual speaker utterance files.

Since the dialogues have two speakers, we assume that the dialogues have been recorded with each speaker on a separate channel. We could break the dialogue into single speaker utterance files by assigning each channel to a single file and transcribe each according to single speaker transcription conventions (ToBI). However, this approach has a serious disadvantage. During a dialogue, usually only one person is speaking at a time. This person speaks for a while, and then the other person takes the turn. Since speakers contribute to the dialogue during their turns, it is essential to capture the sequential aspect of the speakers' turns in the breakup of the dialogue into utterance files. Even breaking a dialogue into speaker turns is not suitable since a speaker might speak for some time, perhaps even a minute or longer before the other person speaks. We feel such long files are awkward for transcribers, and make their task more difficult. We have found that 12 seconds of speech is a good upper bound for most utterance files. So, we propose that dialogues be broken down into utterance files that are at most 12 seconds in length, that preserve local phenomena, and that do not cross speaker turns nor acknowledgments. The result will be a large number of utterance files associated with the dialogue; and so tools are needed to manage these files.

In the rest of this document, we outline the components of our toolkit. These tools are designed to simplify the tasks of setting up the initial dialogue audio file, obtaining an initial breakup of the dialogue, printing the contents of a dialogue, and updating the breakup. We also discuss rules for segmenting the dialogue into utterance files. This is followed by an appendix that gives the man-pages for each tool. The tools and man-pages are available by anonymous ftp transfer as is described in Section 9. These tools assume the presence of the Waves software, available from Entropic Research Laboratories.

2 Guidelines for Segmenting a Dialogue

In order to work with a dialogue, the dialogue needs to be broken up into conveniently sized single-speaker segments so that the sequential aspect of dialogues is captured. In this section, we discuss guidelines that we have found useful for segmenting a dialogue. These guidelines rely on the fact that the two conversants are participating in a dialogue. So, the conversants tend to follow a turn-taking protocol and tend to break their speech into distinct units. Before giving the guidelines, we first briefly review the literature in these two areas.

One of the most obvious features of dialogue is that, for the most part, only one person is talking at a time, thus allowing conversation to be divided into turns. But how do conversants co-ordinate how long a speaker will speak and who will speak next? Sacks, Schegloff and Jefferson (1974) propose that conversants use a locally-managed turn-taking protocol. The basic unit is the *turn-constructional unit* and can be sentential, clausal, phrasal, or lexical constructions (pg. 702). After each turn-constructional unit, there is a *transition relevance place*, where either the current speaker can select the next speaker, as in asking a question to a particular person, or the next speaker can self-select.

However, conversants are not always this well-behaved. Although accounting for no more than 5% of dialogues, there are times when more than one participant is speaking (Levinson, 1983, pg. 296). These might be back-channel responses, such as "okay", or "uh-huh", or the other conversant might be trying to repair an utterance, or he just might be trying to steal the turn, perhaps because it is not necessary for the speaker to finish her thought. So, the guidelines that we propose must be able to deal with such occurrences.

The second issue concerns utterance units. Most theories of discourse, including theories of turn-taking, propose that there is some unit by which the conversational state gets updated. But what is the realization of this unit? This is still an unanswered research question. Previous spoken dialogue research efforts in the TRAINS project have attempted to answer this question. Nakajima and Allen (1993) used four principles: utterance units correspond to sentences of text, they can correspond to basic speech acts, they are at most a single turn, and they can be marked with a pause of at least 750 msec. Gross, Allen, and Traum (1993) proposed that the end of an utterance unit is signaled by one of the following: a boundary tone, a pause in speech longer than a single beat, or a resetting of the pitch level, starting a new intonational phrase.[1] However, giving a definition of utterance units is not one of the aims of the current project. Rather, we feel that where there is ambiguity with a potential utterance boundary, the dialogue should not be segmented there. The boundary should be contained in a single utterance file so that it can be studied and annotated in depth using the Waves software.

[1]See Cruttenden (1986) for an introduction to intonation.

The guidelines that we propose are pragmatically motivated, rather than being motivated by a particular theory of dialogue (namely theories of turn-taking and theories of utterance units). The breakup should allow the dialogue to be used to test such theories. Our segmentation scheme has two aims.

A1: Each utterance file should be short enough so it is easy to analyze (not more than 12 seconds long), and so it does not include effects due to interactions from the other participant.

A2: Each utterance file should be long enough so that local phenomena are not split across utterance-file boundaries.

The first guideline should ensure that the sequence of single-speaker utterance files captures the sequential nature of the dialogue, thus allowing the flow and development of the dialogue to be preserved. In other words, the single-speaker utterance files should not contain or overlap a contribution by the other speaker. The second guideline ensures that the segments allow local phenomena to be easily studied, since they will be in a single file suitable for ToBI annotation. There can be conflicts between these two aims. If this happens, the first aim, (A1), should take priority.

In using the above guidelines for segmenting dialogues, we need to operationalize what constitutes as a suitable place to break up a speaker's speech. We propose the following list of conditions, of which only one needs to hold. These are ordered by the appropriateness of the resulting break.

C1: A suitable break occurs in a speaker's speech whenever she stops and the other speaker starts (or continues), without the first trying to continue.

C2: A suitable break occurs whenever all of the following criteria hold.

1. There is an intonational phrase boundary.
2. There is a major syntactic category (i.e. NP or S) boundary.
3. There is a breath or pause.

C3: A suitable break occurs whenever the first two criteria of (C2) hold, and there is a break between the words (but shorter than a pause).

C4: A suitable break occurs whenever two of the three criteria of (C2) hold.

In segmenting a dialogue, utterance unit boundaries should be chosen so that (A1) and (A2) are satisfied using the strongest breaks possible.

2.1 Examples

For most cases, appropriate places to segment a dialogue will be easy to find. However, difficulties can arise, in which the subjective nature of the above guidelines will become very apparent. So in the following, we give examples that illustrate how to segment such cases. The first is an example of breaking up a long stretch of speech into shorter utterance files. The second is an example of an other-person repair. The next two are examples of where one conversant acknowledges another's utterance before it is even finished. The next two are examples of both speakers competing for the turn. The last is an example of a less structured overlap in speech.

In the examples, we display the proposed segmentation by giving a screen dump of the output of the `showupdate` tool, which is discussed in Section 8.2.[2] In brief, `showupdate` displays the audio file, along with three annotation tiers for each of the two speakers. The first three correspond to the speaker of the top audio file, who plays the role of the user in our TRAINS dialogues, and the last three correspond to the speaker of the bottom audio file, who is the system. The first tier for each speaker contains the word annotations. The second displays the segmentation markers: `<` denotes the beginning of an utterance file and `>` denotes the end. The third tier displays the names of the utterance files. (When using `showupdate`, all of these annotation files are automatically generated; the user simply changes the segmentation tiers. But this is jumping ahead of ourselves.) Also, the time from the beginning of the dialogue is displayed beneath the audio file.

Long Stretches of Speech

Consider the excerpt given in Figure 1, from dialogue d93-21.1.[3] Although this excerpt does not exhibit all of the complications that can arise in segmenting a dialogue, it does illustrate the major points. Since there is no overlapping speech nor other-repairs, we can use condition (C1) to first segment the excerpt based on speaker turns at 2 and 23 sec. Doing this will ensure that requirement (A1) is met.

Next, we notice that the second turn is rather large, about 21 seconds in length. So, (A1) requires us to segment it into smaller parts. It turns out that there are two pauses, both about two seconds in length, after the words "Corning um", and after the phrase "let's see". There is also a pause, about one second in length, after the word "ship". Since "ship" is in the middle of a syntactic unit and does not end an intonational phrase, the pause after it should not be considered as a candidate for an end of an utterance file.

The pause after the phrase "let's see" is a good candidate for condition (C2), for it coincides with an intonational boundary and is not in the middle of a syntactic constituent. Segmenting here will result in two utterance files, both less than 12 seconds in length. Even though the first utterance is less than 12 seconds in length, we could further segment it.

[2]We use `showupdate`, rather than the tool for specifying the initial breakup of the dialogue, `showbreakup` (Section 5.1), because `showupdate` displays the words that have been annotated.

[3]For each excerpt, we give the dialogue name and the utterance file that is is taken from. Dialogue names that start with "d91" are available in Gross, Allen, and Traum (1993), and ones that start with "d92" and "d93" are available in Heeman and Allen (1995).

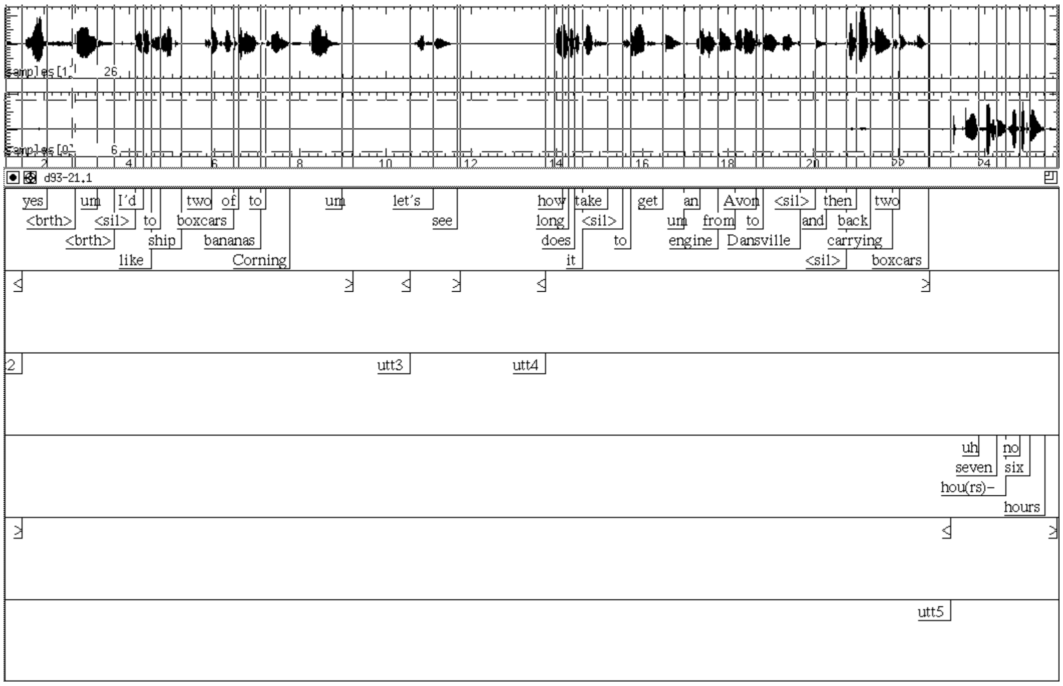

Figure 1: Long Stretch of Speech (d93-21.1)

Two possible points are between the words “Corning” and “um”, and between “um” and “let’s”. The first alternative coincides with an intonational phrase ending and the end of a syntactic constituent. The second co-incides with a large pause. Furthermore, “um”s between utterances do not play a sentential role, nor can they be easily analyzed as being part of an intonational phrase. So, the second alternative is also possible. An overriding advantage of the first is that it would put the “um” with the phrase “let’s see”. Since both are probably filling a similar discourse role, this analysis is preferred.

More important than where the utterance file boundaries are placed is where they are not placed. We definitely do not want a boundary after the word “ship”, even though there is a one second pause there. Nor would we want one anywhere in the region around “get um an engine” especially with the presence of “um”, which is operating locally within the syntactic unit.

Sometimes, there will not be suitable places where a long stretch of speech can be segmented into utterances files that are less than twelve seconds in length. Consider the excerpt given in Figure 2. Although there are many silences in the speech, none of these occur at major syntactic boundaries or at intonational boundaries. Furthermore, one of the silences is at a speech repair (34.0 sec), and the repair involves word correspondences starting with the word “and” at 29.0 sec. and ending with the word “boxcars” at 36.0 sec. So, segmenting anywhere within this range would separate a local phenomenon (Heeman and Allen, 1994). Hence, for this example, we are forced to violate requirement A2, due to the lack of a suitable place that meets one of the conditions C1 through C4.

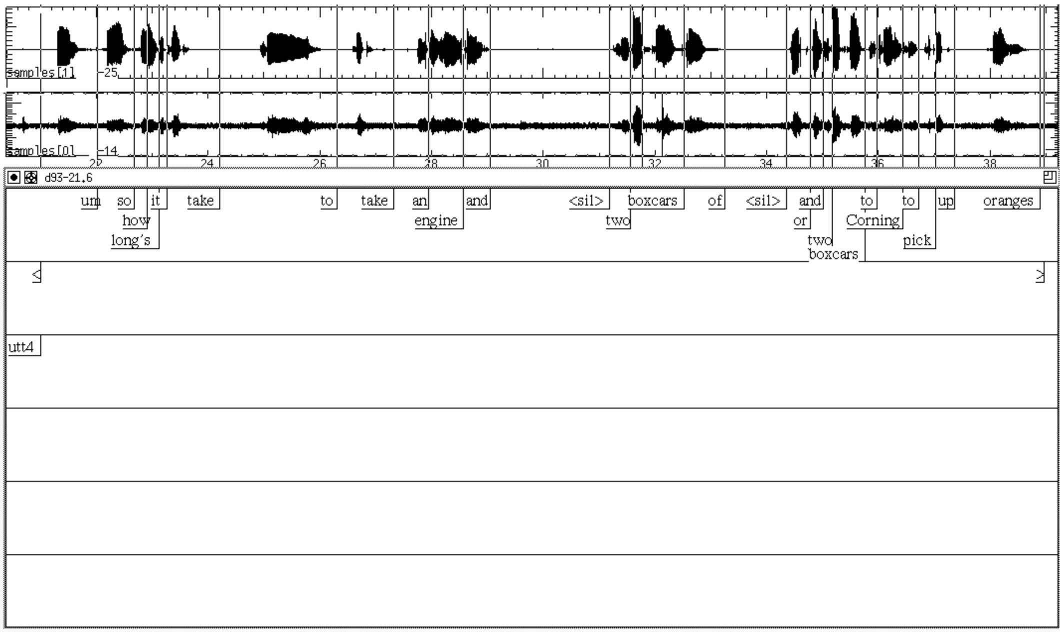

Figure 2: Long Stretch of Speech (d93-21.6)

Other-Speaker Repairs

Other-speaker repairs (repairs made by the hearer) are problematic, for the hearer probably wants to fix the problem immediately, rather than wait until the speaker is willing to cede the turn. So, the hearer interrupts, usually causing the speaker to stop and fix up the problem. So, the speaker’s utterances should be separated into at least two utterance files, one for before the interruption, and one for after.

Consider the example given in Figure 3 (d91-6.1). Here, the system interrupts the user to correct the identity of the engine that is to be used. The user obviously plays attention to this, and in fact repeats the repair and then continues on with the rest of the utterance. So, the part after the utterance of “to” at 43.0 sec. qualifies as a place to break the dialogue by (C1). If we do not segment here it will seem as if the user made a self-repair “engine E to engine E one to Bath”, so would violate requirement (A1).

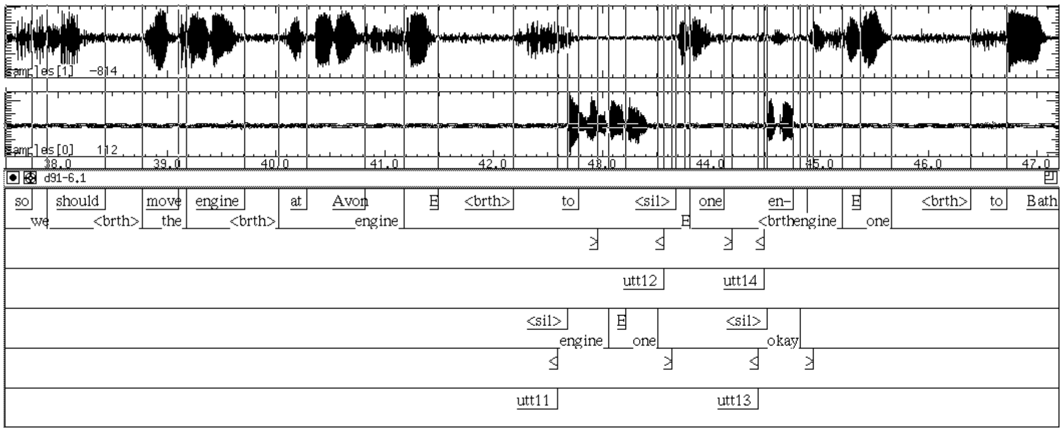

Figure 3: Other-Speaker Repair (d91-6.1)

Overlapping Acknowledgments

Figure 4 is an example of an acknowledgment that seems a bit premature. The user has only uttered “and how many boxcars of oranges”, when the system acknowledges the question. In fact, the system’s “okay”, overlaps with the end of the user’s question “now in Bath”.

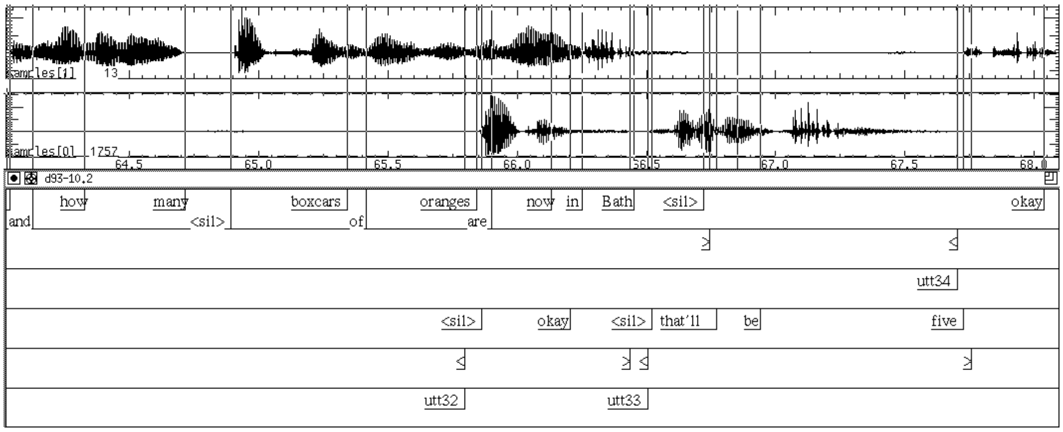

Figure 4: Overlapping Acknowledgment (d93-10.2)

Since the “okay” does not seem to influence the user in finishing his query, we wait until the end of the query to end the utterance file. The next question is whether we should break up the system’s response between “okay” and “that’ll be five”. This break certainly qualifies since it meets all three criteria of (C2) as well as the criteria of (C1). It is also is required by (A1) as well, since the system seems to let the user finish his question, since she waits until the end of “now in Bath” before answering the question. So, we segment here.

Figure 5 gives a second example of an acknowledgment that overlaps the other speaker's speech. The user was asking whether only one engine is needed to transport two boxcars. However, the system understands the question before the user is even finished asking it, and so responds right away. This seems to cause the user to not finish asking the rest of the question. So, the end of the user's partial question qualifies as a place to segment the user's speech by (C1). Also, it is advocated by (A1), since the user's response of "okay" is in response to the system's answer to the partial question.

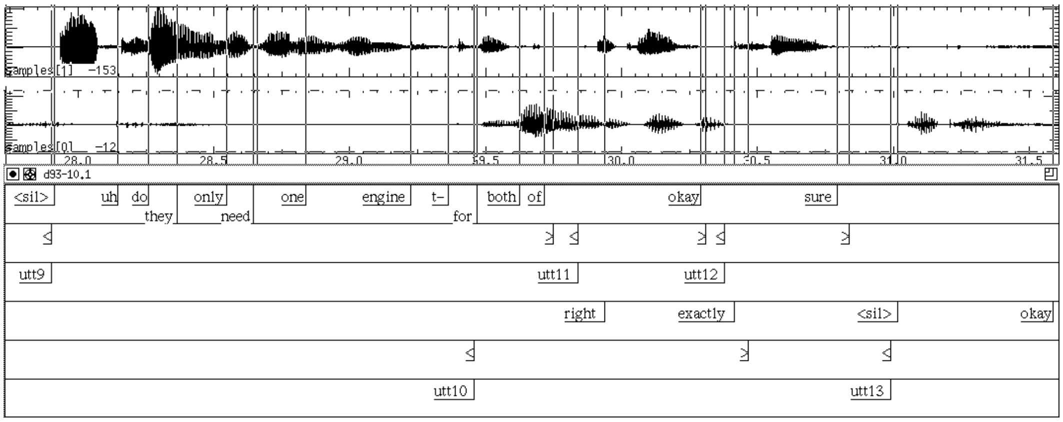

Figure 5: Overlapping Acknowledgment (d93-10.1)

A second issue is whether to segment the system's speech between "right" and "exactly". We choose not to, since the "exactly" does not seem to be in response to the user's "okay", and so not required by (A1). Nor is there a pause or a clear intonational phrase ending in between the two words.

A third issue is whether to segment the user's speech between "okay" and "sure". The "sure" was uttered after the system completed "right exactly", while the "okay" overlapped it. So, requirement (A1) seems to suggest that we need to segment here. Since there is a suitable pause, there is a clear intonation phrase ending on "okay", and there is a major syntactic boundary, the criteria of (C2) are met. So, we segment the user's speech between "okay" and "sure".

Competing for the Turn

Figure 6 illustrates that turn-taking is not always so orderly. In this example the user and system seem to be fighting over the turn. The system finished a complete utterance at 68.0 sec., which the user acknowledged with "okay". The user then tried to assume the turn. But the system wasn't ready to cede it. However, after uttering "so that'll have", the system gives up, and lets the user have the turn.

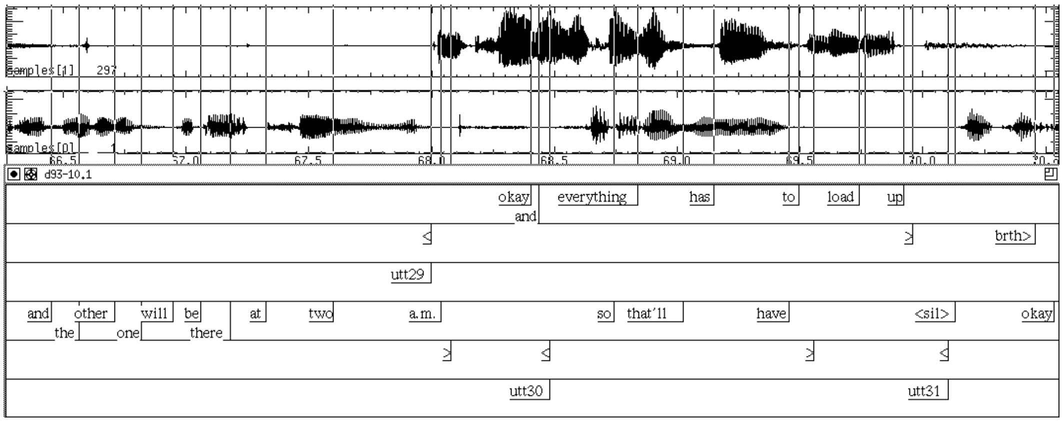

Figure 6: Competing for the Turn (d93-10.1)

First, we mark the end of the system's complete utterance at 68.0 sec, since it meets all three criteria of (C2) as well as the criteria of (C1). It is also advocated by (A1), since the user "okay"s the system's utterance. This means that we need to start an utterance file at the beginning of her attempt to keep the turn at 68.5 sec. The system gives up this attempt after uttering the word "have", and so we mark the end of the utterance file at 69.5 sec. As for the user, his utterance file starts with the word "okay". But how far should this utterance file extent? We could end this file after the "okay", which would capture the sequence of the system attempting to build onto the "okay", and so an utterance file boundary here is advocated by (A1). However, there is no break in the user's speech stream, so a boundary here would only qualify under condition (C4), the lowest ranking of the conditions. But even (C4) requires there to be an intonational phrase ending; from listening to the speech, this does not seem to be present. So, we choose not to segment here.

Figure 7 is another example of the speakers' competition for the turn. The user acknowledges, and tries to clarify the system's contribution, but the system is not finished yet. When he finally does finish, the user grabs the turn and repeats what he said earlier during his attempt to take the turn.

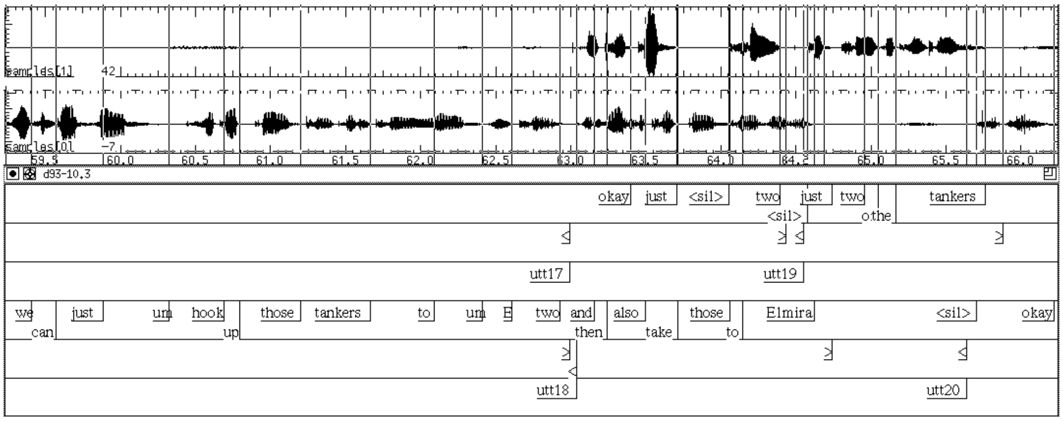

Figure 7: Competing for the Turn (d93-10.3)

Here we segment the system's utterance at 63.0 sec, before they start competing for the turn. Although the pause at this point is very small, it is a clear example of an intonational phrase ending and it is the completion of a major syntactic category, so qualifies by (C3). Also it is needed by (A1) since "okay" seems to be acknowledging the speech that ends at 63.0 sec. We also segment at 64.5 segment in the user's speech, since it seems that she is starting over, due to non-local effect of the system probably not hearing what she just said, and so is advocated by (A1). Condition (C1) seems to allow this to be marked as a break since the user stopped speaking to let the system finish at 64.4 sec.

Overlapping Speech

Figure 8 gives an example in which the user seems to be talking to himself, counting the time it takes to complete the various tasks. It does not seem that the system is influenced by his speech. (The output of `showupdate` here is a bit confusing. The user's speech does not have more amplitude than the system's; it only appears so since the two audio windows are scaled differently.) The first one, "two", seems to be completely ignored by the system, and the system's subsequent speech is not even stressed. So, we do not segment the system's speech at the overlap of "two" at 60.8 sec, since it does not meet any of the conditions for a break nor is it required by (A1).

The next overlap occurs with the system saying "three" followed shortly by the user saying "three". The system's speech can be segmented here since the three criteria of (C2) hold. (It is unclear if (C1) applies.) Also, segmenting here would better help capture the sequencing of speech events, so is advocated by (A1).

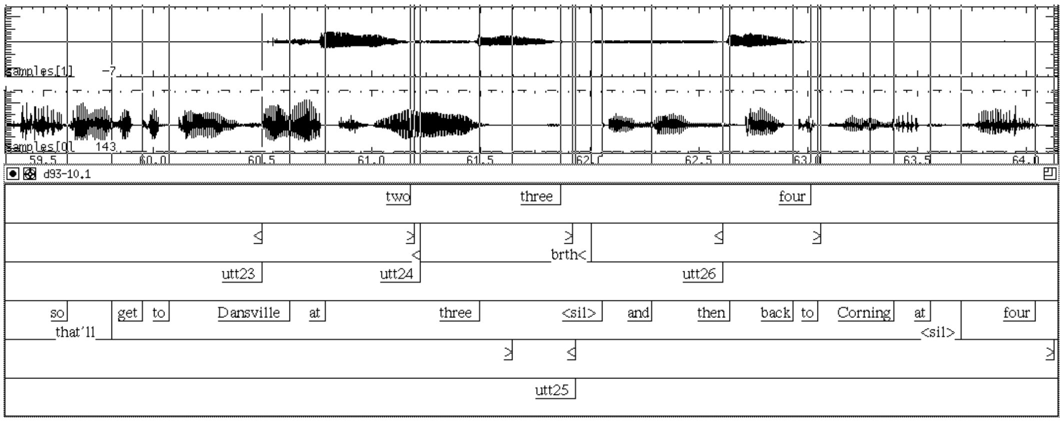

Figure 8: Overlapping Speech (d93-10.1)

The third overlap occurs with the user saying "four". Again the system does not seem to be influenced, and more importantly, there is no appropriate place to break up the system's speech after the overlap. So, we leave the system's utterance of "and then back to Corning at four" as one utterance file.

The last remaining question is how to segment the user's speech. Each of the user's words seem to be expressed in complete intonational phrases, there is a pause between each, and since each is a syntactic fragment, they qualify as major syntactic categories. So, (C2) applies. Furthermore, due to the overlap that occurs with the system's speech, each should be put in a separate utterance file so as to meet requirement (A1).

2.2 Breaths

In a high quality recording of a dialogue, breaths of the speakers can be heard. Often, before a speaker begins an utterance, there will be an audible breath. Since this information might be of importance for analyzing turn-taking and since it is not really part of an utterance, it is best to annotate it at the dialogue level.

The dialogue level transcription involves marking the beginnings and endings of utterances and breaths using the marker `brth<` for starting a breath, `brth>` for ending a breath, `<` for starting an utterance, and `>` for ending an utterance. Since often the ending of a breath coincides with the beginning of an utterance file, the end of the breath should not be marked. Likewise, if the end of an utterance coincides with the beginning of a breath, do not mark the beginning of the breath. For example, a breath that ends at the start of an utterance would be marked as follows.

```
brth<   <                    >
```

An utterance that ends with a breath would be marked as follows.

```
<               >     brth>
```

As a matter of priorities, all breaths immediately preceding an utterance should be marked. Other breaths should be marked unless this becomes too onerous a task. Also, if a breath is between two utterances with a pause on either side, the breath should be marked as belonging to the utterance that follows the breath, as is illustrated below.[4]

```
<                  >    <                 >
               brth<
```

3 Overview of Process

The process of segmenting a dialogue and managing the utterance files starts with the dialogue already recorded on a digital audio tape (DAT), with a separate channel for each speaker. This is important due to the occurrences of overlapping speech and back-channel responses. Without the separate channels, it would be difficult to separate what each conversant is contributing to the conversation. Each dialogue will have its own directory, and the name of the directory is taken as the name of the dialogue. All of the dialogue tools must be run in the dialogue directory.

An overview of the process is given in Figure 9. The first step is to create the dialogue file, `dialog.fea`, from the DAT original. The second step is to break the dialogue into utterance files, which have a base name of `utt`*i*, where *i* is the index of the utterance. At this stage, audio files (`utt`*i*`.fea`) and formant files (`utt`*i*`.f0`) are created for each utterance. The third step is to transcribe each utterance, and this step uses tools and standards for transcribing single speaker utterances. The fourth step comprises a tool to format the utterance level annotation files, so that they can be viewed or printed. The fifth step allows the breakup of the dialogue to be changed. This would be necessary if initial breakup is found unsuitable, for instance, in the case that it separates phenomena that should be in the same utterance file, such as speech repairs. Table 1 gives a complete listing of the permanent files associated with a dialogue and its utterances. These files are discussed below.

4 Creating the Dialogue File

The first step in the dialogue transcription process is the creation of the dialogue file, `dialog.fea`. This file will have two channels of data on it, corresponding to the two channels that the dialogue was recorded with, one for each speaker. From the `dialog.fea` file, two separate files, `speaker0.fea` and `speaker1.fea`, can be created for each of the two speakers by using `demux` (1-ESPS). In order to conserve space, these de-multiplexed files are viewed as temporary files, created when needed.

The reason for providing tools to help in the creation of the `dialog.fea` file is that it is difficult to get the right start and stop times of a dialogue. For instance, when a dialogue

[4]The `brth<` marker is shown on a separate line only to show how close this marker is to the preceding one, since it is the right edge of the token that marks the location of the marker with Waves.

Dialogue Level Files

dialog.fea	Audio file of the dialogue. Contains a channel for each speaker.
speaker*s*.fea	Audio file of the dialogue for each speaker (s is either 0 or 1). To reduce storage requirements, this file is created from `dialog.fea` when needed.
speaker*s*.utts	Annotation files for the utterance boundaries for speaker s.
dialog.cmds	List of commands to break the dialogue into the utterances. Derived from `speakers.utts` by `breakup`.
dialog.data	Concise listing of utterance breakdown giving speaker and start and end times. Derived from `speakers.utts` by `breakup`.
dialog.log	Formatted listing of how the dialogue was broken down into utterances. Includes number of utterances and name of dialogue. Derived from `speakers.utts` by `breakup`.
speaker*s*.words	Annotation file for all of the words in the dialogue for speaker s. Created by `make_speakerfiles` from the `utti.words` files. Used by `showupdate`.
speaker*s*.names	Annotation file with the names of each utterance for speaker s at their start and end points. Created by `make_speakerfiles` from `dialog.data`, and used by `showupdate`.
speaker*s*.display	Annotation file that combines the contents of `speakers.words` and `speakers.names`. Created by `make_speakerfiles` for use by `ptranscriber` and `dtranscriber`.

Utterance Level Files

utt*i*.fea	Audio file for utterance i, created by `dialog.cmds`.
utt*i*.f0	Formant file for utterance i, created by `dialog.cmds`.
utt*i*.words	Word annotation file for utterance i.
utt*i*.tones	Tone annotation file for utterance i. Part of the ToBI annotation scheme.
utt*i*.breaks	Break annotation file for utterance i. Part of the ToBI annotation scheme.
utt*i*.misc	Miscellaneous annotation file for utterance i. Part of the ToBI annotation scheme.

Table 1: Dialogue and Utterance Files

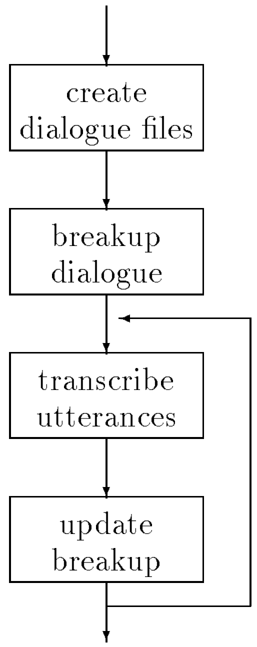

Figure 9: Dialogue Transcription Process

is first recorded, there might be a significant silence before and after the speakers start and finish their conversation included in the audio file on the DAT tape. Also, since the conversation must be downloaded from a DAT drive (or cassette deck), it is difficult to specify the start and end times precisely, assuming that they are even known. In fact, it could take several attempts before a satisfactory ESPS audio file is created, which does not have part of the conversation chopped off nor have an excessive amount of silence (or part of another conversation) at the beginning or end. To alleviate this problem, the tools in this component allow the user to start by creating a `raw` version of the dialogue, with extra time at the beginning and end, and use the Waves software, with its audio and annotation capabilities, to exactly specify the start and end times. Creating the raw version of the dialogue is simple, because the user just needs to make sure that there is an excess at the beginning and end of the dialogue.

Figure 10 gives a flowchart of the tools in this component. The first tool is `dat2raw`, which is used to create the raw dialogue file, `dia.raw`. The second step is `showraw`, which uses Waves to let the user delimit the actual dialogue. The third step is `cutupraw`, and it uses ESPS utilities to create the delimited dialogue, `dialog.fea`. The last step, `cleanupraw`, deletes the temporary files that were created, including the raw dialogue. Below we give an overview of each tool.

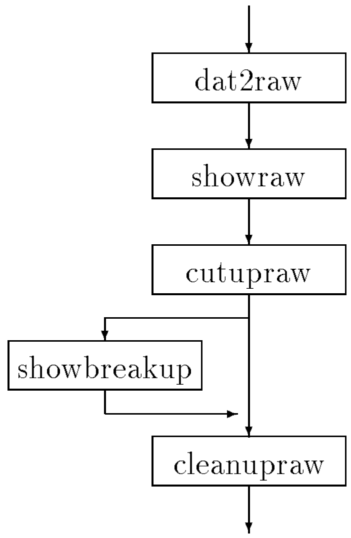

Figure 10: Creating the Dialogue

4.1 dat2raw

The dat2raw program is used for downloading a dialogue from a DAT drive. This program makes use of the DAT-Link software and hardware (Tow, 1992), which allows a DAT player to be accessed as a computer device. The user simply positions the DAT drive at the beginning of the dialogue, or a bit before it, puts the DAT drive into remote mode, and enters the command dat2raw. This program makes use of narecord to record the dialogue into an ESPS headerless audio file. Once the dialogue has finished, simply stop the program with control 'C'. The resulting raw dialogue is called dia.raw.

4.2 showraw

In the previous step, dat2raw was used to download a dialogue from a DAT drive. However, there might be extraneous parts at the beginning and end of the dialogue, perhaps even part of another dialogue. So, the second step, showraw, is used to mark the actual beginning and end of the dialogue. This step makes use of Waves to display the dialogue file, along with an annotation file, dia.edge. The annotation file is where the beginning and end of the dialogue are marked, with the symbols < and >, respectively. These symbols can be accessed from the menu associated with the annotation file. The start and stop marks are used in the next step to cut off the extraneous beginning and end of the raw dialogue.

4.3 cutupraw

After the beginning and end of the raw dialogue have been marked, cutupraw should be run to remove the extraneous beginning and end portions. This program uses the ESPS utility `copysd` to copy the selected portion of the raw dialogue, and the combination of `bhd` and `btosps` to set the start time of the newly created dialogue file to zero. The result will be the `dialog.fea` file. At this point, before the raw file is deleted by the next step `cleanupraw`, `showbreakup` (described in Section 5.1) can be run to verify that the dialogue has in fact been properly delimited.

4.4 `cleanupraw`

The `cleanupraw` tool is the final step in creating the dialogue file. This tool erases all of the extraneous files, including the original raw file. The only remaining files should be `dialog.fea`, the 2 channel audio file corresponding to the dialogue.[5]

5 Breaking up the Dialogue

Section 2 discussed guidelines that we have found useful for breaking up dialogues. In this section, we discuss the tools that help in obtaining the breakup. The breakup is specified through the use of the Waves software in two annotation files, one for each speaker—`speaker0.utts` and `speaker1.utts`. The beginning and end of each utterance file is specified by the symbols `<` and `>`, respectively. These two annotation files are the essence of the breakup and should not be deleted.

Figure 11 gives a flowchart of the tools in this section. The first tool is `showbreakup` and it uses Waves to allow the user to mark the breakup. The second tool is `breakup` and it creates dialogue level files as well as the utterance level audio files and formant files.

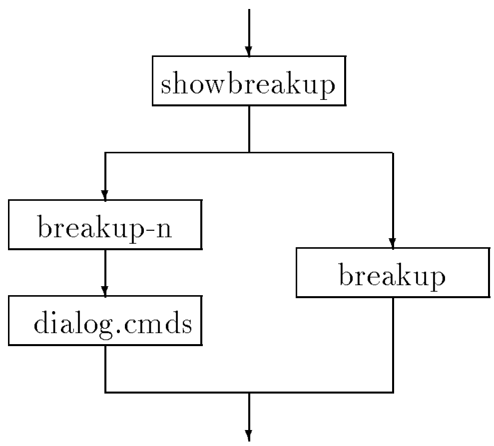

Figure 11: Breaking up the Dialogue

A strategy that we have found useful is to do a "rough" breakup initially. Then, after the words have been transcribed, the breakup can be revised, using the tools given in Section 8.

[5] If `showbreakup` was run, there will be two additional files, `speaker0.utts` and `speaker1.utts`.

5.1 showbreakup

The specification of the breakup of a dialogue is done using the tool `showbreakup`. This tool uses the Waves software to display the dialogue, and two annotation files, `speaker0.utts` and `speaker1.utts`, in which the utterances for each speaker can be marked. In the update step, we will see that these files allow the breakup to be revised.

5.2 breakup

The tool `breakup` has three functions. The first is to ensure that the utterance annotation does not contain an error. For instance, two start utterance markers cannot occur consecutively without an end utterance marker in between. Using the `-c` option will invoke only the error checking part of `breakup`.

The second function is to create dialogue level files that contain information about the breakup. The file `dialog.data` stores the start and stop times for each utterance, as well as the name of the speaker. The file `dialog.log` stores the same information as `dialog.data`, but in a more readable format. It also contains the number of utterances and the dialogue name (name of the directory). The files `dialog.data` and `dialog.log` are accessed by other tools, such as `transcriptor` (Section 7), that need this information. The third file is `dialog.cmds`, and it is a list of the commands, suitable for executing, that will create all of the utterance level files from `dialog.fea`. If the `-n` option is specified to `breakup`, it will just create the dialogue level files (after checking the breakup annotation for errors).

The third function of `breakup` is to create the actual utterance level files. These files are the audio files, `utt`*i*`.fea`, and the formant files, `utt`*i*`.f0`, one of each for each utterance in the breakup. The utterance level files are created by the script `dialog.cmds`. If the `-n` option is specified to `breakup`, then `breakup` will not create the utterance level files, and so the user can execute `dialog.cmds` at a later time.

The tool `breakup` creates the utterance level files ordered by the time at which the utterance files start. If there are two utterance files that start at the same time, the one for `speaker0` is ordered first. So, by manipulating starting times of utterance files (via `showbreakup` or `showupdate`), utterance files from overlapping speech can be ordered appropriately. Consider the excerpt given in Figure 7. Here, the user (top) and system (bottom) both start speaking at 63.0 sec. To order the user's utterance before the system's, the starting time for the user's utterance file was made a bit before the system's. The ordering of the utterance files affects the manner in which the `transcriptor` (Section 7) formats the dialogue.

6 Transcribing Utterances

The main tool that we use for transcribing utterances, `transcriber`, is very similar to the script provided with Waves. However, in transcribing a dialogue, it is cumbersome to list all of the utterance files that are to be transcribed, especially since there are typically more than a hundred utterance files in a five minute dialogue. So, we have added a more convenient way to specify the list of utterances. This functionality is provided by `expand_utts`, and can even be added to an existing `transcriber` script (see Section 9.1).

Also included are two other tools, `scatter` and `dtranscriber`. The former assists in doing word transcriptions, and the latter is a variation of the `transcriber` script that displays the dialogue context of the utterance that is being transcribed.

6.1 `expand_utts`

This utility provides a simple way to specify a list of utterance, for it allows the user to specify *ranges* of utterances. Simply specify the starting utterance by number or basename, followed by a dash, followed by the ending utterance. For instance, '10-20' specifies the range of utterances `utt10` through `utt20`. Also, the symbols '$' and '*' can be used to specify the last utterance and all utterances, respectively.[6] The utterance specification is then converted into a list of the actual utterance basenames, suitable for use with the `transcriber` script.

6.2 `transcriber`

This program is a version of the script that is supplied with the Waves software for creating annotation windows for words, break indices, tones, and miscellaneous items. The script has been enhanced by incorporating a call to `expand_utts`, which simplifies the specification of lists of utterances. It also allows the user to specify the annotation tiers with the `-e` option.

6.3 `scatter`

The tool `scatter` is useful for speeding up word transcription, in cases in which an automatic aligner (e.g., Ent, 1994) is not available. With this tool, the user first does a *rough* transcription, in which individual words are not time-aligned. Rather, words are entered as a group at single time-points. The tool *scatter* will then break apart these groups and spread the individual words evenly between the previous annotation and the time-point of the current group of words. The user, then as the third step, time-aligns the individual words using the `move` facility in Waves. Since the user can concentrate on transcription or word alignment, the time required for completing the overall task is greatly reduced.

If an automatic aligner is available, it should be used in place of `scatter`. To get a copy of the rough transcription, which is to be aligned, use `transcriptor` (see Section 7) with the `-w` option, to get just the word listing.

6.4 `ptranscriber`

The `ptranscriber` script is a variation of `transcriber`, but adapted so that it displays the dialogue context for the utterance that is being displayed. In addition to displaying the

[6] When using '$' or '*', the utterance specification needs to be enclosed in single quotes "'" to prevent interpretation by the shell.

utterance level audio file, pitch contour, and annotation files, `dtranscriber` displays the two channel audio file, the utterance breakup and word annotations for the dialogue. The utterance breakup and word annotations are in two annotation tiers, one for each speaker. These two tiers are for display only; all changes to the annotations must be done at the utterance level.

6.5 dtranscriber

The `dtranscriber` script is yet another variant of `transcriber`. Like `ptranscriber`, it allows the user to see the dialogue context. However, rather than specifying all utterances that are to be transcribed in advance, and having to process them in that order, `dtranscriber` comes with a side menu that that gives the user the option of moving to the next or preceding utterance file, and for jumping to an arbitrary utterance file (by placing the left marker just after its starting time in the dialogue display). The menu also includes options for scattering or aligning the current utterance (the aligning option makes use of the Aligner program (Ent, 1994)). However, this additional functionality results in slower response times for moving from one utterance to the next.

7 Printing the Transcription

When annotating a single utterance, the result can be viewed by invoking `transcriber`. However, when multiple utterances comprise a dialogue, it is often useful to get a printout of the annotations for the entire dialogue. The `transcriptor` program does just that. It prints the annotation files for a dialogue so that the tone, break indices, and miscellaneous items are time-aligned with the word annotations.

Rather than forcing transcribers to mark overlapping speech in an annotation tier, `transcriptor` marks such occurrences automatically with the `+` indicator (it can be disabled with the `-o` flag). Its algorithm for doing this is rather simple, if two utterances overlap in time, the extent of the overlap is marked at the closest word to the overlap. The start and stop times of an utterance are determined by looking at the first non-silence or breath annotated in the words tier[7] and the end of the last word in the words tier.

7.1 transcriptor

The `transcriptor` tool produces a formatted listing of the utterance annotations for the dialogue. The annotations in the tone tier, the break tier, and the miscellaneous tier are time-aligned with the word annotations. The printing of the annotations for these tiers can be disabled by specifying the flag `-t` for tones, `-b` for breaks, or `-m` for miscellaneous. In addition, a single utterance can be printed by specifying the basename of the utterance. Note that this utterance does not have to be part of a dialogue, and so `transcriptor` can be used as a general tool for printing ToBI annotated speech files. Table 2 on page 22 gives the output of `transcriptor` on a sample dialogue.

[7] To make the overlap detection more exact, beginning silences in utterances should be marked with `<sil>` or `<brth>`, as appropriate.

8 Updating the Breakup

In transcribing the utterances of a dialogue, it might be the case that the breakup splits phenomena that should have been included in the same utterance and so the breakup will need to be revised. The tools in this component allow the breakup to be modified without losing the utterance transcriptions.[8]

In Figure 12, the full updating process is given. The process starts with `startupdate`, which sets up a subdirectory, `new`, for creating the new breakup. The second step is `showupdate` and it uses Waves to display the dialogue file so that the user can specify the new breakup. The `showupdate` tool is similar to `showbreakup` except that it displays a copy of the word annotations and the utterance numbering. The third step is `update`. This program compares the old breakup with the new one and creates the new utterance files, including the annotation files, in the subdirectory `new`. If any of the annotations are not accounted for by the new breakup, `update` gives an error message and aborts. In this case, the user either can rerun `showupdate` to make the breakup account for these annotations, or can use `transcriber` to fix the annotation files (in the dialogue directory, not in the `new` directory). After `update` is run with no unaccounted-for annotations, the last step, `installupdate`, can be run. This tool is used to install the new breakup on top of the old dialogue. In the sections below, we discuss each tool in detail. An extra program, `cancelupdate`, allows an update process to be canceled.

8.1 startupdate

The first program in the component is `startupdate`. It creates the subdirectory `new` and copies the dialogue level files into it. It then creates two files, one for each speaker, which contain a numbering of the original utterances. These two files are displayed by `showupdate` in order to allow the user to more easily determine the new breakup. (Annotation files for the transcribed words are also displayed by `showupdate`; however, these are created by `showupdate`.)

8.2 showupdate

The second program in the update process is `showupdate`. This tool, briefly discussed in Section 2.1, uses Waves to display the current breakup, as well as the word annotations and the numbering of the original utterances. The user can then simply change the breakup annotation. Note that the word annotations should not be changed, nor should the utterance numberings. These files are only for display and are not used by in the creation of the new dialogue breakup. If the word annotations need to be changed, simply exit out of `showupdate` and run `transcriber` (in the dialogue directory) on the appropriate utterance. After fixing the annotations, `showupdate` can be rerun, without needing to redo `startupdate`. The word annotations displayed by `showupdate` will reflect the changes just made.

[8] This tool also allows the initial segmentation to be a 'rough draft.' Once all of the utterance files have been annotated, the update tools can be used to fix up the segmentation.

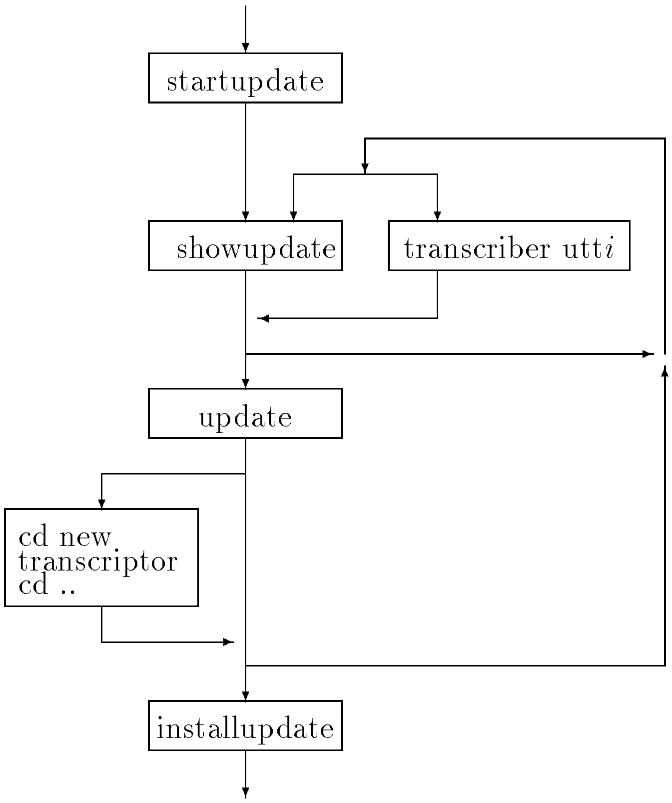

Figure 12: Updating the Breakup

8.3 `update`

The third program in the update process is `update`. This program first creates all of the dialogue level and utterance level files, in accordance with the new breakup in the subdirectory `new`. After running this program, the results can be checked by running `transcriptor` in the subdirectory `new`.

In creating the new breakup, rather than starting from scratch, `update` compares the new breakup, in `new/dialog.data` with the old one, in `dialog.data`. This comparison is done by `uttsdiff`, which outputs a list of commands indicating which utterances need to be renumbered, which ones need to be created, which ones need to be deleted, and which ones remain unchanged. These commands are interpreted by `update`. The annotations for the utterances that need to be deleted are dumped into temporary files, one for each speaker and type of annotation (see `dump_utts`), for instance `new/update.speaker0.words`, and are used for creating the annotations for the utterances that need to be created (see `make_utts`). If any of the annotations in the temporary files are not accounted for by the utterances that are created, an error message is given that lists the unaccounted for annotations and the utterance files that they originated from. If this happens, the user will need to resolve the problem either by changing the new breakup with `showupdate` or by fixing the original utterances with `transcriber` (in the dialogue directory).

The set of annotation tiers that `update` will process can be specified using the `-e` option. Any annotation tiers not in this list will be lost for utterances that need to be deleted and created (e.g., due to changing the size of an utterance). For utterances that need to be renumbered, all files that begin with the utterance basename are renamed appropriately.

8.4 `installupdate`

The fourth program in the update process is `installupdate`. This program installs the new version of the dialogue from the subdirectory `new` and deletes the `new` subdirectory. This step can only be run if no problems were encountered with `update`.

8.5 `cancelupdate`

The fifth program, `cancelupdate`, cancels an update that has not yet been installed. Since all files affected by the update process are kept in the subdirectory `new`, this program simply removes all of the files from that directory and removes the directory.

8.6 An Example

To illustrate how the update process works, we use dialogue d93-121.1, the dialogue used for Figure 1. Table 2 gives the output of `transcriptor` on the dialogue, with the segmentation that we suggested for it.

```
utt1  : s:  hello can I help you
utt2  : u:  yes um I'd like to ship two boxcars of bananas to Corning
utt3  :     um let's see
utt4  :     how long does it take to get um an engine from Avon to
            Dansville and then back carrying two boxcars
utt5  : s:  uh seven hou- no six hours
utt6  : u:  okay <sil> six hours and then how long <noise> to fill
            the <sil> two boxcars with bananas and ship it to Bath
utt7  : s:  uh five hours
utt8  : u:  okay let's do that
```

Table 2: Breakup of Dialogue

Let's say that we decided that the third and fourth utterances should be joined. First, we would go to the dialogue directory and run `startupdate`. This will create the subdirectory `new` as well as the files needed for displaying the dialogue with Waves. Second, we run `showupdate`. This will bring up Waves on the dialogue audio file, along with six annotation tiers. The first one gives the word annotations that have been transcribed for `speaker1`, the next the breakup, and the third the names of the utterances. The first and third tiers allow us to easily find the places in the dialogue that we need to change (changes to these two tiers will have no effect on the update process). Continuing on, the fourth through the sixth tiers are the same as the first through third, but for `speaker0`. So, if we want to merge the 3rd and 4th utterance, we simply locate the markers that denote the end of utterance 3 and the beginning of utterance 4, and delete them. We then exit from Waves.

After `showupdate`, we run `update`. This script will determine that for the new breakup the third utterance has changed, and so a new feature file must be created for it, as well as the appropriate annotation files. It will also determine that the fourth utterance of the old breakup has been removed, and so it needs to renumber the rest of the utterances. By the end of this process, a new copy of all of the utterance level files will be created in `new`. Lastly, we run `installupdate`, which will update the dialogue directory with the new breakup.

9 Installing the Tools

The dialogue tools make use of `csh`, `awk`, `sed` and a C-compiler, which are all widely available in UNIX environments. They also make use of the Waves (TM) software, by Entropic Research Laboratories. The tools can be retrieved by anonymous ftp transfer from `ftp.cs.rochester.edu`, and are located in `pub/packages/dialog-tools` in the compressed tar file `toolset.tar.gz`.[9] This file can be uncompressed with `gunzip toolset.tar`, and unpacked with `tar -xvf toolset.tar`. This will create four subdirectories: `doc`, `man`, `src`, `bin`. A copyright notice, a copy of the instructions for installing, and notes about the current release will be in `doc`; The man pages for the tools will in `man`; the source for all of the C programs in `src`; the `csh` scripts in `bin`; and the menus for the Waves tools in `bin/menus`. At this point, it is necessary to `make` all of the C programs into executables. Simply run "`make all`" in the `src` subdirectory. This will compile all of the C programs, and move the executables to `bin`.

Next, the two environmental variables `PATH` and `MANPATH` must be changed so as to include the `bin` and the `src` subdirectory, respectively. These variables are usually defined in your `.cshrc` or in your `.login` file.

The next step is to set the location of the Waves menus. The scripts that invoke Waves have a variable called `MENUS`. This variable must be set to the absolute name of the subdirectory `bin/menus`. It is currently set to `/s9/dialogs/bin/menus`, the location where it is installed on our local system, but this will undoubtedly need to be changed. So, you must edit the scripts `showraw`, `showbreakup`, `showupdate`, `transcriber`, `ptranscriber`, `dtranscriber`, and `dfunctions`.

The next step is to decide the sampling frequency of the audio files. The default value is 16000 kHz. To change this, simply edit the scripts `dat2raw`, `showraw`, and `cutupraw` and change the value of the assignment to the variable `FREQ`.

The next step is to customize `dat2raw`. This program, as delivered, makes use of DAT-Link software and hardware (Tow, 1992) to interface to a DAT machine. The program

[9] If you have access to a WWW viewer, visit our home page at `http://www.cs.rochester.edu/research`.

that it uses is narecord, which records the two channel audio signal into a audio file. The dat2raw script can be easily customized to use site specific software or hardware.

The last step involves finding a program that will play the audio files on your computer. Waves uses the global variable play_prog for specifying this. The scripts showraw, showbreakup, showupdate, transcriber, ptranscriber, and dtranscriber currently have this set to s16play, which sends the audio file to the Sun Sparc audio port. See (Ent, 1993) for other options. If the default needs to be changed, change the following line in the scripts.

```
echo 'set play_prog s16play' >> $TMP
```

The above scripts also support playing sound to a DAT machine. For this we use the program naplay, which interfaces to the DAT machine through DAT-Link. This option is available through the side menu audio for showraw, showbreakup, showupdate and transcriber, and the side menu dtools for dtranscriber. These menus can be left as is, or altered appropriately to support other options.

9.1 Adding expand_utts to a transcriber script

Included in the installation is a transcriber script. Since there might already be a transcriber script that has been locally customized, it might be best to simply add the dialogue support for specifying ranges of utterances to the existing script. This is a rather simple customization. Simply determine the manner in which the utterance list is currently being specified. This is probably through the assignment of some variable, say UTTS, with the arguments to the script, which for a csh script would be $argv. Simply change this so that the result of expand_utts on the arguments is assigned instead. For a csh script, this would be the following.

```
set UTTS=`expand_utts "$argv"`
```

Note that expand_utts is downward compatible, and so can take as input a list of utterances separated by spaces, and will only interpret tokens consisting of just a number. Hence, it can be used with non-dialogue utterance files.

10 Acknowledgments

We wish to thank Bin Li and Tsuneaki Kato for their help in formulating an earlier version of the heuristics for breaking up a dialogue. We also thank Bin for his help in developing the initial version of some of the tools. We thank Andrew Simchik for his help in making this technical note more readable and easier to understand, and for his help in refining and writing some of the tools. We also thank David Traum for his comments on our segmentation guidelines.

References

Allen, James F. and Lenhart K. Schubert. 1991. The TRAINS project. Technical Report 382, Department of Computer Science, University of Rochester, May.

Beckman, Mary E. and Gayle M. Ayers. 1994. Guidelines for ToBI labelling, version 2.0. Manuscript and accompanying speech materials, Ohio State University, (obtain by writing to tobi@ling.ohio-state.edu).

Beckman, Mary E. and Julia Hirschberg. 1994. The ToBI annotation conventions. Manuscript, Ohio State University, (obtain by writing to tobi@ling.ohio-state.edu).

Cruttenden, Alan. 1986. *Intonation.* Cambridge: Cambridge University Press.

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

Entropic Research Laboratory, Inc., 1994. *Aligner Reference Manual.* Version 1.3.

Gross, Derek, James Allen, and David Traum. 1993. The Trains 91 dialogues. Trains Technical Note 92-1, Department of Computer Science, University of Rochester, June.

Heeman, Peter and James Allen. 1994. Detecting and correcting speech repairs. In *Proceedings of the 32th Annual Meeting of the Association for Computational Linguistics*, pages 295–302, Las Cruces, New Mexico, June.

Heeman, Peter A. and James Allen. 1995. The Trains 93 dialogues. Trains Technical Note 94-2, Department of Computer Science, University of Rochester.

Levinson, Stephen C. 1983. *Pragmatics.* Cambridge University Press.

MADCOW. 1992. Multi-site data collection for a spoken language corpus. In *Proceedings of the DARPA Workshop on Speech and Natural Language Processing*, pages 7–14, February.

Nakajima, Shin'ya and James F. Allen. 1993. A study on prosody and discourse structure in cooperative dialogues. Trains Technical Note 93-2, Department of Computer Science, University of Rochester, September.

Sacks, Harvey, Emanuel A. Schegloff, and Gail Jefferson. 1974. A simplest systematics for the organization of turn-taking for conversation. *Language*, 50(4):696–735, December.

Silverman, K., M. Beckman, J. Pitrelli, M. Ostendorf, C. Wightman, P. Price, J. Pierrehumbert, and J. Hirschberg. 1992. ToBI: A standard for labelling English prosody. In *Proceedings of the 2nd International Conference on Spoken Language Processing (ICSLP-92)*, pages 867–870.

Townshend Computer Tools, 1992. *DAT-Link User's Manual.* Revision 2.11.

A Manual Pages for Tools

One-line summary of Tools

```
breakup (1) - break dialog into individual utterances.
cleanupraw (1) - remove temporary files from creation of dialog
cutupraw (1) - create dialog file by cutting up raw dialog
dat2raw (1) - download a dialog from the DAT-drive
dtranscriber (1) - display utterance and dialogue for annotation
installupdate (1) - copy updated breakup into dialog file
ptranscriber (1) - display utterance so that it can be annotated
scatter (1) - scatter word transcriptions evenly over utterance
showbreakup (1) - display dialog so that utterances can be marked
showraw (1) - display raw dialog so start/end can be specified
showupdate (1) - display dialog so user can update its breakup
startupdate (1) - setup a subdirectory for update to use.
transcriber (1) - display utterance so that it can be annotated
transcriptor (1) - format dialog transcription files
update (1) - re-generate utterance files from new breakup
```

NAME

breakup - break dialog into individual utterances.

SYNOPSIS

breakup [-c -n -h]

DESCRIPTION

Breakup is used to break the dialog file into individual utterances as specified in speaker0.utt and speaker1.utt, which were created by ‘showbreakup’.

OPTIONS

-c Only check the breakup files to ensure their correctness.

-n Create the dialog level files, but do not create the utterance level files, the .fea and .f0 files. Since all commands needed to create the utterance level files are in ‘dialog.cmds’, they can be created later by executing ‘dialog.cmds’.

-h Give help information.

USAGE

This tool must be run after ‘showbreakup’. Creating the individual utterance files can take an extended amout of time, so be prepared to wait. Or run it with the -n flag and run the resulting ‘dialog.cmds’ in batch.

FILES

The following files are input to breakup.

dialog.fea	Audio file for dialog.
speaker0.utts	Utterance boundaries for speaker 0.
speaker1.utts	Utterance boundaries for speaker 1.

The following files are created by breakup.

dialog.cmds	List of commands to create the utterance files.
dialog.data	Listing of utterance start/stop times and speaker.
dialog.log	Information about the dialog.
utti.fea	Audio file for utterance i.
utti.f0	Formant file for utterance i.

SEE ALSO

showbreakup(1), update(1).

BUGS

No bugs currently known to exist.

AUTHOR

Bin Li (binli@cs.rochester.edu).

Peter A. Heeman (heeman@cs.rochester.edu).

NAME

cleanupraw - remove temporary files from creation of dialog

SYNOPSIS

cleanupraw

DESCRIPTION

Cleanupraw gets rid of the temporary files that were creating in setting up the dialog files.

USAGE

Cleanupraw should be run after 'cutupraw' has cut the raw file down to size.

FILES

The following files are deleted:

dia.raw	Raw dialog file.
dia.edge	Annotation file with start/stop markers.

DIAGNOSTICS

If 'dia.raw' or 'dia.edge' do not exist, then the program aborts.

SEE ALSO

dat2raw(1), showraw(1), cutupraw(1).

AUTHOR

Peter A. Heeman (heeman@cs.rochester.edu).

NAME

cutupraw - create dialog file by cutting up raw dialog

SYNOPSIS

cutupraw

DESCRIPTION

Cutupraw cuts the raw dialog to its proper start and end times from the information specified by the user in ‘showraw’.

USAGE

Cutupraw should be used after ‘showraw’ to cut the dialog file down to the right size. It uses the information in dia.edge to determine the start and end times. After using-ing this tool, the user can run ‘breakup’ to ensure that the dialog has been properly cut.

FILES

The following files are used as input by cutupraw:

dia.fea	Raw dialog file, with header, that is to be cut up.
dia.edge	Annotation file with the start/stop markers.

The following files are work files created by cutupraw:

dia1.fea	Cut up dialog file, with non-zero start time.
dia2.raw	Beheaded version of dia1.fea

The following file is created by cutupraw:

dialog.fea	Audio file for the dialog.

DIAGNOSTICS

If ‘dia.fea’ or ‘dia.edge’ doesn’t exist, then the program aborts.

SEE ALSO

dat2raw(1), showraw(1), cleanupraw(1).

AUTHOR

Peter A. Heeman (heeman@cs.rochester.edu).

NAME

dat2raw - download a dialog from the DAT-drive

SYNOPSIS

dat2raw

DESCRIPTION

Dat2raw downloads a dialog from the DAT-drive. The dialog is put into the current directory under the name ‘dia.raw’. This is a headerless ESPS file.

USAGE

Dat2raw is used to download a dialog from the DAT-drive into the file ‘dia.raw’. Simply position the DAT-drive at the start of the dialog (make sure you are a few seconds before the first words are uttered) and put the DAT into ‘remote’ mode. (The ‘pause’ and ‘rewind to start of track’ are useful for positioning the DAT drive.) Then run this program. Once the dialog has finished, press ‘cntl-c’. It doesn’t matter if part of the next dialog is recorded before ‘cntl-c’ is pressed. Then, ‘showraw’ can be used to display the dialog and to mark the actual beginning and end.

FILES

dia.raw Raw dialog file downloaded from DAT drive.

DIAGNOSTICS

If dia.raw already exists, then the program aborts.

SEE ALSO

showraw(1), cutupraw(1), cleanupraw(1).

AUTHOR

Peter A. Heeman (heeman@cs.rochester.edu).

NAME

dtranscriber - display utterance and dialogue for annotation

SYNOPSIS

dtranscriber [-e tiers] utterance

DESCRIPTION

This script is a variation of transcriber, but adapted so that it can display the dialogue context as well as the individual utterances. In addition to displaying the audio file, pitch contour, and annotation files for an utterance, it also displays the audio file and special annotation files for the dialogue. The dialogue annotation files, one for each speaker, are for display only and consist of entries that mark the start and end of each utterance, and entries for the words that were spoken.

The dtranscriber script also comes with a side menu that gives options for moving to the next or preceding utterance file, for jumping to an arbitray utterance file (by placing the left marker just after its starting time). The menu also includes options for scattering or aligning the current utterance (the aligning option makes use of Entropic's Aligner program).

This program should be used after the dialogue has been broken down into utterance files.

OPTIONS

-e tiers

Display annotation files with extensions in the quoted list tiers. The default is "misc breaks words tones". Note that the order that the annotation tiers is listed in will be opposite the order which they are ordered on the screen.

USAGE

This tool needs to be run in the same directory in which the utterance files are located. The first utterance to be transcribed is passed as the only argument to dtranscriber.

Do not use the 'continue' button for moving to the next utterance. Instead use the 'next' button on the side menu.

FILES

```
     dialog.log       List of utterances.  Used to determine number
                      of utterances for interpretation of '$' and '*'.
```

DIAGNOSTICS

The file "dialog.log" must be present in the current directory in order to use the meta-symbols '$' and '*'.

SEE ALSO

ptranscriber(1), transcriber(1), breakup(1), transcriptor(1), expand_utts(1).

AUTHOR

Peter A. Heeman (heeman@cs.rochester.edu).

Andrew D. Simchik (simchik@cs.rochester.edu).

NAME

installupdate - copy updated breakup into dialog file

SYNOPSIS

installupdate [-e tiers]

DESCRIPTION

This program copies the updated breakup into the dialog file and cleans up all files created by the update process. The updated breakup is copied from the 'new' subdirectory, and once this is done, everything in 'new' is deleted.

OPTIONS

-e tiers

Since some dialogues might have more utterance level files then the 'csh' can handle at once, 'installupdate' moves the utterance level files by file extension. The default list of extensions is "misc breaks words tones". Note that 'installupdate' moves all files from the 'new' subdirectory, so this option does not need to be specified under normal use.

USAGE

This is the last of the four programs for updating the breakup of a dialog, and so should be run after 'startupdate', 'showupdate', and 'update'. This program should be run in the directory of the dialog, and copies the new breakup from the subdirectory 'new' into the dialog directory.

DIAGNOTICS

This program will only execute if it finds the file update.done in the subdirectory 'new'. This file acts as a flag to indicate that 'update' was successful.

BUGS

Dialogues with large number of utterance files can cause problems. For these, make sure that you specify all utterance extensions that you use, so that the program will have a better chance of moving the utterance level files.

SEE ALSO

startupdate(1), showupdate(1), update(1).

```
AUTHOR
     Peter A. Heeman (heeman@cs.rochester.edu)
```

NAME

ptranscriber - display utterance so that it can be annotated

SYNOPSIS

ptranscriber [-e tiers] utterance-list

DESCRIPTION

This script is a variation of transcriber, but adapted so that it can display the dialogue context as well as the individual utterances. In addition to displaying the audio file, pitch contour, and annotation files for an utterance, it also displays the audio file and special annotation files for the dialogue. The dialogue annotation files, one for each speaker, are for display only and consist of entries that mark the start and end of each utterance, and entries for the words that were spoken.

This program should be used after the dialog has been broken down into utterance files.

OPTIONS

-e tiers
: Display annotation files with extensions in the quoted list tiers. The default is "misc breaks words tones". Note that the order that the annotation tiers is listed in will be opposite the order which they are ordered on the screen.

USAGE

This tool needs to be run in the same directory in which the utterance files are located. The list of utterances to be transcribed is passed as the only argument to transcriber. The utterance-list is a list of utterance specifications, separated by commas. Utterance specifications can be one of the following. may include:

n Utterance file with base name uttn. (Can also be referred to by uttn.)

$ The last utterance. (Can also be referred to by utt$.)

* All utterances. (Can also be referred to by

utt*.)

n-m An inclusive range of utterance files, where n and m refer to specific utterances.

If the utterance list includes the metasymbols '$' or '*', then it should be enclosed with single quotes, "'", to avoid interpretation by the shell.

EXAMPLES

The following illustrate some examples of using the utterance list with transcriber.

transcriber '*'
To transcribe all utterances.

transcriber 'utt10-$'
To transcribe from utterance 10 through to the end of the dialog.

transcriber '10-20,$'
To transcribe from utterance 10 through to 20, and the last utterance.

FILES

dialog.log List of utterances. Used to determine number of utterances for interpretation of '$' and '*'.

DIAGNOSTICS

The file "dialog.log" must be present in the current directory in order to use the meta-symbols '$' and '*'.

SEE ALSO

transcriber(1), dtranscriber(1), breakup(1), transcriptor(1), expand_utts(1).

AUTHOR

Peter A. Heeman (heeman@cs.rochester.edu).

NAME

scatter - scatter word transcriptions evenly over utterance

SYNOPSIS

scatter utterance-list

DESCRIPTION

Scatter is a tool to speed up word transcribing. It takes the word transcriptions, in which multiple words have been transcribed for the same time point, and spreads them evenly between the previous time-aligned transcription and the time point of the current annotation. The scattering process then makes it easier for a user to hand-align the individual words, since these words just need to be re-aligned. This tool is only useful if an automatic aligner is not available.

USAGE

This tool works with ‘xlabel’ annotation files which end with the extension ‘.words’. The user should first ‘roughly’ transcribe the words in the audio file by placing multiple words at the same ‘xlabel’ time point. By not time-aligning each word, nor putting each word in a separate ‘xlabel’ time point, the transcription process is greatly speeded up (by about a factor of 6!). After the rough transriptions have been done, execute ‘scatter’ with the basename of the file(s) (see ‘expand_utts’ for how lists and ranges of files can be specified). Then, use ‘xwaves’ to time align each word.

Note that only time-points with more than one token will be scattered. Also, more than one file can be scattered at a time by specifying an utterance-list, which is a list of utterance specifications, in which ranges of utterances can be specified. See ‘expand_utts’ for more details. Note that scatter will rewrite the ‘.words’ file with the scattered transcription, saving the pre-scattered version in ‘.wordsBAK’.

DIAGNOSTICS

The file "breakup.data" must be present in the current directory in order to find the start time of an utterance. The file "dialogue.utts" must be present in the current directory in order to use the meta-symbols '$' and '*".

SEE ALSO

expand_utts(1)

BUGS

No bugs currently known to exist.

AUTHOR

Bin Li (binli@cs.rochester.edu)

Andrew Simchik (simchik@cs.rochester.edu).

NAME

showbreakup - display dialog so that utterances can be marked

SYNOPSIS

showbreakup

DESCRIPTION

Displays the dialog.fea file so that the user can break it up into speaker utterance files.

USAGE

This tool displays a Waves annotation window so that the user can annotate the beginning and ending of each utterance. These markers are saved in speaker0.utt and speaker1.utt. The audio file for the dialog as well as the separate ones for each speaker are displayed so that the user can listen to both speakers combined or to each individually.

It is important that the files ‘speaker0.utts’ and ‘speaker1.utts’ not be deleted for they specify how the utterance files relate to the dialog, and they allow any utterance level annotations to be saved if the breakup is revised through ‘update’.

FILES

dialog.fea	Audio file for dialog.
speaker0.fea	Audio file for speaker 0.
speaker1.fea	Audio file for speaker 1.
speaker0.utts	Utterance boundaries for speaker 0.
speaker1.utts	Utterance boundaries for speaker 1.

SEE ALSO

generate(1).

AUTHOR

Bin Li (binli@cs.rochester.edu).

Peter A. Heeman (heeman@cs.rochester.edu).

NAME

showraw - display raw dialog so start/end can be specified

SYNOPSIS

showraw

DESCRIPTION

Showraw uses Waves to display the raw dialog in the current directory that was just retrieved from the DAT drive. It lets the user mark the beginning and the end of the actual dialog, so that it can be cut up with ‘cutupraw’.

USAGE

This program should be used after a dialog has been downloaded from the DAT drive. This is because when downloading a dialog, it is not possible to be accurate in so far as the starting and ending times go. So, ‘showraw’ should be used to mark the actual beginning and end of the dialog. The program ‘cutupraw’ should be run next to create the actual dialog.

FILES

The following file is input to showraw.

dia.raw	Raw dialog file from the DAT drive.

The following file is created by showraw.

dia.fea	Raw dialog file, with header, for use with Waves.
dia.edge	Annotation file with start/stop markers.

DIAGNOSTICS

If no dia.raw file exists, the program aborts with an error message.

SEE ALSO

dat2raw(1), cutupraw(1), cleanupraw(1).

AUTHOR

Peter A. Heeman (heeman@cs.rochester.edu).

NAME

showupdate - display dialog so user can update its breakup

SYNOPSIS

showupdate

DESCRIPTION

This program uses waves to show the current breakup of the dialog, so that the user can change it. To simplify this task, the word transcriptions are shown, as well as the utterance names (created by 'make_speakerfiles').

USAGE

This program must be run after 'startupdate' and before 'update'.

FILES

The following list of files are displayed by waves.

new/dialog.fea	Audio file of the dialog.
new/speaker0.fea	Audio file of speaker 0.
new/speaker1.fea	Audio file of speaker 1.
speaker0.words	Word annotations of speaker 0.
speaker0.words	Word annotations of speaker 1.
speaker0.names	Utterance numbering of speaker 0.
speaker0.names	Utterance numbering of speaker 1.

The following files are updated by the user using waves.

new/speaker0.utts	Utterance breakup for speaker 0.
new/speaker1.utts	Utterance breakup for speaker 1.

SEE ALSO

startupdate(1), update(1), make_speakerfiles(1).

BUGS

No bugs currently known to exist.

AUTHOR

Peter A. Heeman (heeman@cs.rochester.edu)

NAME

startupdate - setup a subdirectory for update to use.

SYNOPSIS

startupdate

DESCRIPTION

This program makes and sets up the subdirectory 'new', which 'showupdate' and 'update' will use for updating the breakup. 'startupdate' gathers all of the word transcriptions and numbers all of the utterances so that this information can be shown by 'showupdate'.

USAGE

This program is the first of the series of programs to run when the breakup of a dialog needs to be updated (the others are 'showupdate' and 'update'). It must be run in the directory of the dialog. But note, it does all its work in the subdirectory 'new'.

FILES

The following files are copied into the subdirectory 'new',

dialog.fea	Audio file of dialog.
speaker0.fea	Audio file of speaker 0.
speaker1.fea	Audio file of speaker 1.
speaker0.utts	Utterance boundaries for speaker 0.
speaker1.utts	Utterance boundaries for speaker 1.

SEE ALSO

showupdate(1), update(1).

BUGS

No bugs currently known to exist.

AUTHOR

Peter A. Heeman (heeman@cs.rochester.edu)

NAME

transcriber - display utterance so that it can be annotated

SYNOPSIS

transcriber [-e tiers] utterance-list

DESCRIPTION

Transcriber is a script similar to the one supplied with Waves and is used for annotating utterance files. It uses the Waves software to display the utterance file, and puts up a window for the user to transcribe the words, tones, breaks, and miscellaneous items. It takes a single argument, a list of utterances, which the Waves software will be invoked on.

This program should be used after the dialog has been broken down into utterance files.

OPTIONS

-e tiers
: Display annotation files with extensions in the quoted list tiers. The default is "misc breaks words tones". Note that the order that the annotation tiers is listed in will be opposite the order which they are ordered on the screen.

USAGE

This tool needs to be run in the same directory in which the utterance files are located. The list of utterances to be transcribed is passed as the only argument to transcriber. The utterance-list is a list of utterance specifications, separated by commas. Utterance specifications can be one of the following. may include:

- n — Utterance file with base name uttn. (Can also be referred to by uttn.)
- $ — The last utterance. (Can also be referred to by utt$.)
- * — All utterances. (Can also be referred to by utt*.)

n-m An inclusive range of utterance files, where n and m refer to specific utterances.

If the utterance list includes the metasymbols '$' or '*', then it should be enclosed with single quotes, "'", to avoid interpretation by the shell.

EXAMPLES

The following illustrate some examples of using the

utterance list with transcriber.

transcriber '*'
    To transcribe all utterances.

transcriber 'utt10-$'
    To transcribe from utterance 10 through to the end of the dialog.

transcriber '10-20,$'
    To transcribe from utterance 10 through to 20, and the last utterance.

FILES

dialog.log   List of utterances. Used to determine number of utterances for interpretation of '$' and '*'.

DIAGNOSTICS

The file "dialog.log" must be present in the current directory in order to use the meta-symbols '$' and '*'.

SEE ALSO

ptranscriber(1), dtranscriber(1), breakup(1), transcriptor(1), expand_utts(1).

AUTHOR

Bin Li (binli@cs.rochester.edu).

Peter A. Heeman (heeman@cs.rochester.edu).

NAME

transcriptor - format dialog transcription files

SYNOPSIS

transcriptor [-b] [-m] [-o] [-r] [-s] [-t] [-p] [-T] [-R] [-w]
[-W times audio-file] [utterance-name]

DESCRIPTION

Transcriptor produces a transcription of the dialog that is in the current directory. It puts the word transcriptions, tone transcriptions, and repair transcriptions on different lines, and aligns them. If an utterance file is specified, than just that one utterance is printed. Note that transcriptor can be used for non-dialog utterance files.

OPTIONS

-b Ignore the utt*.break files. When this option is not specified, the break information is used to augment the tone information for deducing intermediate and intonational phrase endings.

-m Ignore the utt*.misc files. Hence, repair annotations will not be printed out. When used in conjunction with the -e option, the .misc files are read in, but not printed out.

-o Don't check for overlapping utterances. When this option is not specified, '+' signs are used to mark overlapping speech. The start and end times of an utterance are compared with the times of other utterances to see if there is an overlap. If there is, '+' marks where the other utterance begins and ends with respect to the words in the current utterance.

-s When not specified, the initial and final silences and breaths in an utterance file are removed. This makes the calculation of overlapping utterances more precise, since beginning and ending silences are not counted.

-t Ignore the utt*.tone files. When this option is not specified, the tones that were transcribed are printed underneath the corresponding word transcriptions.

-p Process the utt*.pos files. When this option is specified, the part-of-speech tags that were transcribed are printed underneath the corresponding word transcriptions.

-T For each turn (change of speaker), print "<turn>". This option is useful in combination with the -w option to make a file with the <turn> breaks inserted.

-w Print a listing of the word transcriptions only, removing all word tokens that start with '<'. This listing is suitable for doing word counts or for obtaining frequency counts.

USAGE

The various options allow different information to be given in the printout. With no options specified, it formats all of the information, and indicates overlapping utterances. To just get the word transcriptions, use the options '-m -b -t -o'.

Note that break information is used to determine the phrase boundaries. Word boundaries are taken to define the end of the word (not break indices). Tones, breaks, and miscellaneous tokens that are a bit past the end marker are taken to belong to the preceding word.

DIAGNOSTICS

Transcriptor parses the tonal information. If it cannot parse the tone, it will give an error message. At most three tones are allowed per word.

SEE ALSO

transcriber(1).

AUTHOR

Peter A. Heeman (heeman@cs.rochester.edu).

NAME

update - re-generate utterance files from new breakup

SYNOPSIS

update [-e tiers]

DESCRIPTION

This program re-generates the dialog utterance files, given a new breakup for them. Update does the new breakup, preserving any previous transcriptions that were already done for the dialog. The new utterance files are created in the sub directory ''new''.

OPTIONS

-e tiers

Update annotation files with extensions in the quote list tiers.Thedefaultismisc breaks words tones tiers not in this list will be lost for utterances that need to be deleted and created (e.g., due to changing the size of an utterance). For utterances that only need to be renumbered, all files that begin with the utterance basename are renamed appropriately.

USAGE

This is the third program in the process for updating the breakup of a dialog, and so should be run after 'startupdate' and 'showupdate', and before 'installupdate'. This program should be run in the directory of the dialog. It compares the original breakup with the new one, and creates and changes any files, as necessary. If no problems are encountered with running this program, then 'installupdate' can be run to install the new breakup.

DIAGNOSTICS

If the new breakup results in some annotations falling between utterances, then an error message will be given, and the list of unaccounted-for annotations will be stored in the file update.error. If this case arises, run showupdate to fix the breakup, and then run update again.

SEE ALSO

startupdate(1), showupdate(1), installupdate(1).

AUTHOR

Peter A. Heeman (heeman@cs.rochester.edu)

B Manual Pages for Subroutines

One-line summary of Tools

```
dump_utts (1) - dump annotations for utterances
expand_utts (1) - expand utterance list into list of filenames
find_speaker (1) - determines the speaker for each utterance
make_speakerfiles (1) - create speaker level files suitable for displaying
make_utts (1) - make annotations for utterances
strip_header (1) - strip off header of an annotation file
uttsdiff (1) - find differences between two dialog breakups
```

NAME

dump_utts - dump annotations for utterances

SYNOPSIS

dump_utts [-e extensions] utterance_list speaker0 speaker1

DESCRIPTION

dump_utts dumps all of the annotations for utterances specified by utterance_list into files prefixed by speaker0 and speaker1.Thereasonfortwo annotations for the two speakers separate. Otherwise, especially in the case of overlapping speech, their words will become interwined. Speaker information comes from the 'dialog.data' file.

OPTIONS

-e extensions

Only dump annotations with an extension in extensions. Members of extensions are separated by spaces (to group all of the extensions as one parameter, double-quotes should be placed around the list). If not specified, it will default to "words misc tones breaks".

FILES

dialog.data List of utterances and their speakers.

SEE ALSO

make_utts(1), update(1).

AUTHOR

Peter A. Heeman (heeman@cs.rochester.edu).

NAME

expand_utts - expand utterance list into list of filenames

SYNOPSIS

expand_utts utterance-list

DESCRIPTION

This tool takes an utterance-list specification and expands it into a list of the actual utterance files, separated by commas. This program makes the specification of a list of utterances much simplier.

USAGE

This program is intended to be used by 'transcriber' to expand the utterance list that the user specifies to a list of utterance files. The utterance list is the only argument to 'expand_utts'. The utterance-list is a list of utterance specifications, separated by commas. Utterance specifications can be one of the following. may include:

n Utterance file with base name uttn. (Can also be referred to by uttn.)

$ The last utterance. (Can also be referred to by utt$.)

* All utterances. (Can also be referred to by utt*.)

n-m An inclusive range of utterance files, where n and m refer to specific utterances.

If the utterance list includes the metasymbols '$' or '*', then it should be enclosed with single quotes, "'", to avoid interpretation by the shell.

'expand_utts' is downward compatible with the way in which utterance lists are specified in the 'transcriber' script that is supplied with Waves. So, spaces are allowed to delimit utterances, and non-dialog utterances can be specified.

EXAMPLES

The following illustrate utterance list specifications.

```
'*'  To specify all utterances.

'utt10-$'
     To specify utterance 10 through to the end of the  dia-
     log.

'10-20,$'
     To specify utterance 10 through to  20,  and  the  last

     utterance.
```

FILES

```
dialog.log       List of utterances.  Used to determine number
                 of utterances for interpretation of '$' and '*'.
```

DIAGNOSTICS

The file "dialog.log" must be present in the current directory in order to use the metasymbols '$' and '*'.

SEE ALSO

transcriber(1).

AUTHOR

Peter A. Heeman (heeman@cs.rochester.edu).

NAME

find_speaker - determines the speaker for each utterance

SYNOPSIS

find_speaker [-s speaker] annotation_files

DESCRIPTION

This program takes a list of utterance file basenames (separated by spaces, with no quoations around the list) and outputs speaker of each one to standard output. Speaker information is found by looking in 'dialog.data'.

OPTIONS

-s speaker

Output the utterance file basenames that were spoken by speaker where speaker is either 'speaker0' or 'speaker1'.

USAGE

This program is used by dump_utts to determine the speaker of the utterance files that it is to dump, so that it can keep the annotations for each speaker separate.

SEE ALSO

dump_utts(1).

AUTHOR

Peter A. Heeman (heeman@cs.rochester.edu).

NAME

make_speakerfiles - create speaker level files suitable for displaying

SYNOPSIS

make_speakerfiles

DESCRIPTION

This program creates the speaker-level annotation files (speaker?.words, speaker?.names, speaker?.display) that can be used when viewing the entire dialogue with WAVES. The 'speaker?.words' files contain all of the transcribed words (generated from utt?.words); the 'speaker?.names' files contain markers for the start and end of each utterance (labeled by 'utt?<' and 'utt?>' respectively) and are generated from the 'speaker?.utts' files; and the 'speaker?.display' files combine the 'speaker?.words' and 'speaker?.names' into a single file for each speaker, so as to reduce the number of annotation tiers needed to display the dialogue information.

USAGE

This program is automatically run by 'installupdate' and 'showupdate' (showupdate uses the speaker level files for display purposes). When transcribing a dialogue, or making changes to an existing one, the user might want to run this script in order to get the new word transcriptions displayed on the dialogue tiers (see 'ptranscriber' and 'dtranscriber').

FILES

The following files are created.

speaker0.words	Word annotations for speaker0.
speaker1.words	Word annotations for speaker1.
speaker0.names	Utterance boundaries for speaker0.
speaker1.names	Utterance boundaries for speaker1.
speaker0.display	Combined word and utterance boundaries for speaker0.
speaker1.display	Combined word and utterance boundaries for speaker0.

SEE ALSO

showupdate(1), dtranscriber(1), ptranscriber(1).

BUGS

No bugs currently known to exist.

AUTHOR

Peter A. Heeman (heeman@cs.rochester.edu)

NAME

make_utts - make annotations for utterances

SYNOPSIS

make_utts [-e extensions] utterance-list speaker0 speaker1

DESCRIPTION

make_utts makes annotation files for the utterances specified by utterance-list from the files prefixed by speaker0 and speaker1.

FILES

dialog.data	Breakup of dialog into utterances and by speaker. Needed in order to determine which speaker file each utterance should be dumped to.

SEE ALSO

dump_utts(1), update(1).

AUTHOR

Peter A. Heeman (heeman@cs.rochester.edu).

NAME

strip_header - strip off header of an annotation file

SYNOPSIS

strip_header [-e extension] annotation_files

DESCRIPTION

This program takes a list of annotation files (separated by spaces, with no quoations around the list) and outputs the labels, with the headers stripped out, to standard output.

OPTIONS

-e extension
: Annotation files are specified by just their base name. Append extension to the base names.

USAGE

This program is used by dump_utts to remove the headers off of annotation files, and to concatenate them together.

SEE ALSO

dump_utts(1).

AUTHOR

Peter A. Heeman (heeman@cs.rochester.edu).

NAME

uttsdiff - find differences between two dialog breakups

SYNOPSIS

uttsdiff newbreakup oldbreakup

DESCRIPTION

uttsdiff compares two dialog breakups, newbreakup and oldbreakup. The breakups are the dialog.data files produced by 'breakup'. The differences are written in the form of commands for how to create the new annotation files for the new breakup from the old breakup. The commands are written to standard output.

USAGE

This program is used by 'update' in order to determine how to create the new utterance files.

SEE ALSO

update(1), generate(1).

AUTHOR

Peter A. Heeman (heeman@cs.rochester.edu).