

ACT: A GRAPHICAL DIALOGUE ANNOTATION COMPARISON TOOL

Fan Yang

Susan E. Strayer

Peter A. Heeman

Computer Science and Engineering
OGI School of Science and Engineering
Oregon Health & Science University, USA

yangf@cse.ogi.edu susan_strayer@yahoo.com heeman@cse.ogi.edu

ABSTRACT

Although there are a number of tools for annotating dialogue, little work has been done in visualizing differences among annotations. Visualization of differences in annotation should help in doing consensus annotation, refining an annotation scheme and training novice annotators. In this paper, we present a graphical Annotation Comparison Tool (ACT), which displays multiple annotations side by side and highlights the differences. The differences our tool currently captures are utterance segmentation, utterance order, speech repairs and utterance categories.

1. INTRODUCTION

There is much interest in annotating human-human dialogue. Annotated dialogues are useful for formulating and verifying theories of dialogue and as training data for statistical models. A number of dialogue annotation tools have been developed for facilitating the work of annotators, examples of which can be found at the Linguistic Data Consortium (www ldc.upenn.edu).

Annotating dialogues is difficult. Different people might have different analyses for a piece of dialogue, and even the same person might have different analyses at different times. Hence, in annotating a corpus, it is imperative to evaluate intercoder reliability to assess the quality of annotations or an annotation scheme. There are statistical comparison packages for calculating intercoder reliability, using statistical measures such as alpha, kappa [2] or percent agreement. However, there are other aspects of annotation than intercoder reliability, where other tools could help.

We propose a tool for visually comparing annotations. The tool is useful for doing consensus annotation, which reduces coding errors. To reach a consensus, annotators need to compare and resolve their differences. A graphical tool can facilitate this process by highlighting the location and nature of their differences and by showing the context in which the differences occur, which should allow the annotators to decide which annotation is correct.

A second reason for visually comparing annotations is to improve an annotation scheme. By carefully examining the differences among multiple annotators, an annotation scheme can be improved by clarifying the border between annotation categories, or by collapsing or splitting annotation categories. An annotation scheme is a trade-off between usefulness and consistency of coding [3]. If in most cases, when one annotator tagged a piece of speech as category ‘A’ and another annotator coded it as ‘B’, it means that the distinction between ‘A’ and ‘B’ is unclear, and we

may need to either specify the difference between ‘A’ and ‘B’ more clearly, or we may need to merge categories ‘A’ and ‘B’. In addition to using statistical techniques to determine which categories are unclear, a graphical tool is useful for examining specific instances.

Thirdly, a graphical comparison tool can be used for instructing novice annotators. Annotators are usually trained by giving them an annotation scheme, having them do a sample dialogue, and comparing their annotation with that of an expert. With a tool that highlights the location and the nature of the differences between their annotations and the expert’s, trainees can more easily master the annotation scheme.

In this paper, we present a graphical dialogue annotation comparison tool, the Annotation Comparison Tool (ACT), which visually shows the differences between two or more annotations. In Section 2, we discuss the first level of comparison, which is based on utterance segmentation, utterance order and speech repairs. In Section 3, we discuss our comparison of utterance categories. Section 4 gives a full description of ACT. We conclude with a description of future work in Section 5.

2. UTTERANCE IDENTITY

After the words of a dialogue have been transcribed, the next step is splitting the speech of the two speakers into a sequence of utterances. The definition of an utterance will come from the annotation scheme being used. Any definition that requires an utterance to be a grouping of consecutive words by a single speaker will work with our tool. We also include speech repair annotations, since they are used to determine the speaker’s intended utterances. This section discusses what it means for utterances to be identical and how we identify regions where utterances are not the same.

2.1. Utterance segmentation

Coders may segment utterances differently based on different analyses of the dialogue. Consider Example 1, which is based on an excerpt from the Trains corpus [5]. One possible segmentation

system:	take the remaining boxcar down to	Corning			
user:	right to du-	Corning			
time (s)	76	77	78	79	80

Example 1

is Segmentation 1-1 below. A basis for this segmentation is that utterance u1-1 begins to complete s1-1 and s2-1 finishes the completion. Utterance u2-1 is uttered as a repetition of s2-1.

System (s1-1): take the remaining boxcar down to
 User (u1-1): right to du-
 System (s2-1): Corning
 User (u2-1): Corning

Segmentation 1-1

Another segmentation of Example 1 is Segmentation 1-2 below, where the analysis is that both u1-2 and s2-2 complete s1-2 and that they overlap each other (as is indicated by the '+' signs in s2-2).

System (s1-2): take the remaining boxcar down to
 User (u1-2): right, to du- Corning
 System (s2-2): +Corning+

Segmentation 1-2

Utterances that do not have the same word content are considered to be different, as seen in Segmentations 1-1 and 1-2.

2.2. Utterance ordering

Our tool takes as input utterance ordering. Although an utterance ordering in an annotation can be derived from the begin time of the first word in the utterance, this order might not reflect how the dialogue is developing [7]. Hence our comparison tool takes this as a separate input and visualizes differences in utterance ordering. Consider Example 2 below.

system:				ok
user:	pick up the three boxcars	that	takes	three hours
time (s)	29	30	31	32 33

Example 2

An order based on the begin time of the first word in each utterance is given in Segmentation 2-1. In this segmentation, it seems that “ok” is an acknowledgement to “that takes three hours”, unless one realizes that the “ok” overlaps u2-1.

User (u1-1): pick up the three boxcars
 User (u2-1): that takes three hours
 System (s1-1): ok

Segmentation 2-1

Another ordering is given in Segmentation 2-2. Here, the system’s “ok” has been moved up so that it only acknowledges “pick up the three boxcars”, but does not acknowledge “that takes three hours”. The fact that “ok” overlaps u2-2 is no longer significant in the utterance ordering.

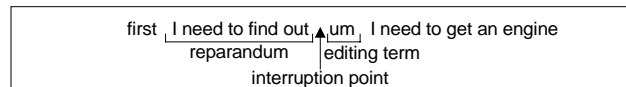
User (u1-2): pick up the three boxcars
 System (s1-2): ok
 User (u2-2): that takes three hours

Segmentation 2-2

Utterances that do not have the same previous context are considered to be different, as seen in Segmentations 2-1 and 2-2.

2.3. Speech repairs

People often begin speaking before they are sure of what they want to say, and so speakers might go back and repeat or modify what they just said, as illustrated by Example 3. Speech repairs are common occurrences in spoken dialogue. In the Trains Corpus, 23% of speaker turns contain at least one repair and 54% of turns with at least ten words contain a repair [6].



Example 3

A speech repair is composed of three parts: reparandum, interruption point and editing term [6]. The *reparandum* is the speech that is being replaced, the *interruption point* occurs at the end of the reparandum, and the *editing term* consists of words such as “um”, “uh”, “ok”, and “let’s see” that help signal the repair. Heeman & Allen [6] classified speech repairs into three types: fresh starts, modification repairs, and abridged repairs. The intended utterance for Example 3 is: “first I need to get an engine”. Different speech repair annotations lead to different intended utterances.

2.4. Identical utterances

If one is building a scoring tool, one should score at the most fine-grained level possible. In the two segmentations of Example 1, such a tool might score utterances s1-1 and s1-2 as being identical and utterances s2-1 and s2-2 as also being identical. Utterances u1-1, u2-1 and u1-2 would be scored as different.

Segmentation 1-1	Segmentation 1-2
s1-1	s1-2
u1-1	u1-2
s2-1	s2-2
u2-1	

A scoring tool’s view of different segmentations

However, the purpose of our visual comparison tool is to draw attention to regions where differences exist. Our tool would also highlight utterances s2-1 and s2-2, including them in one large highlighted block as shown below. We intend that the highlighted block contains all of the differences the annotators need to rectify at a time.

Segmentation 1-1	Segmentation 1-2
s1-1	s1-2
u1-1	u1-2
s2-1	s2-2
u2-1	

Our view of different segmentations

To find blocks of different utterances we first define *identical utterances*. To say two (or more) utterances are identical, we mean they have the same content and the same previous context. The *content* of an utterance is the set of word tokens in the utterance. The *previous context* of an utterance is the set of word tokens in all of the utterances previous to it. *Word token* refers to a word spoken by a certain person at a certain time. Our definition of identical utterances captures the phenomena of utterance segmentation and ordering. By this definition, utterances s2-1 and s2-2 in Example 1 are not identical, since the set of word tokens occurring prior to s2-1 is not the same as those occurring prior to s2-2.

A stricter definition of identical utterances includes identical speech repair tags. There are several levels of speech repairs

comparison. From the weakest to the strongest level of comparison, they are:

- The utterances contain the same word tokens after removing the reparandum and editing terms in each utterance.
- The reparandum, interruption points, and editing term tags are the same, in addition to the above.
- The speech repair types are the same, in addition to the above.

3. UTTERANCE CATEGORY COMPARISON

Once we have identified identical utterances, we can also check whether they have the same categories. The categories used are determined by the researcher to meet the needs of the research project. For example, categories could consist of speech acts, such as Statement, Info-request, Answer and Acknowledgement. Traditionally, speech act theory attempts to capture the purpose of an utterance purpose with one label. Researchers are now developing annotation schemes that allow multiple labels in multiple layers to be applied to an utterance, such as DAMSL [4]. Our tool accommodates such annotation schemes.

Our tool allows minor distinctions in utterance categories to be ignored by specifying equivalence classes. This is especially useful in the course of developing an annotation scheme, as it allows the annotators to focus on different aspects. For example, the difference between Acknowledgement and Agreement can be ignored by specifying the rule <Backard.Acknowledgement> == <Backward.-Agreement>. Another example in DAMSL is to have all forward functions in one category equivalence class and all backward functions in another class, so the researcher can focus only on the different annotations between forward or backward functions.

When coding multiple layers, there are usually interactions among them which are worked out during development of an annotation scheme. When certain conditions arise, we allow coding differences to be ignored. For instance, when Correct_Misspeaking is coded by both annotators, differences in coding Statement can be ignored.

4. ACT

4.1. Interface

ACT has a graphical front-end that visually presents the differences between two or more annotations as shown in Figure 1. Similar to our dialogue annotation tool [7], each annotation is shown as a sequence of utterances with their utterance categories. Blank lines are inserted to keep identical utterances from different annotators aligned. This is shown in figure 1, where a blank line is inserted into the display of the second annotator's utterances to align the identical "ok"s. A single scrollbar on the right controls views of all annotations simultaneously.

Differences in utterance annotation are highlighted to allow coders to focus on significant annotation differences. Red signals a difference in utterance content or prior context; orange signals differences in speech repairs; and yellow signals that the corresponding utterances are different only in categories. The level of speech repair comparison is specified by the user. Differences that are not significant can be unhighlighted manually so that users can focus more on meaningful distinctions. Users can add rules to the configuration file to specify their own category equivalence classes for their annotation scheme.

The utterance by utterance view hides a lot of the details of

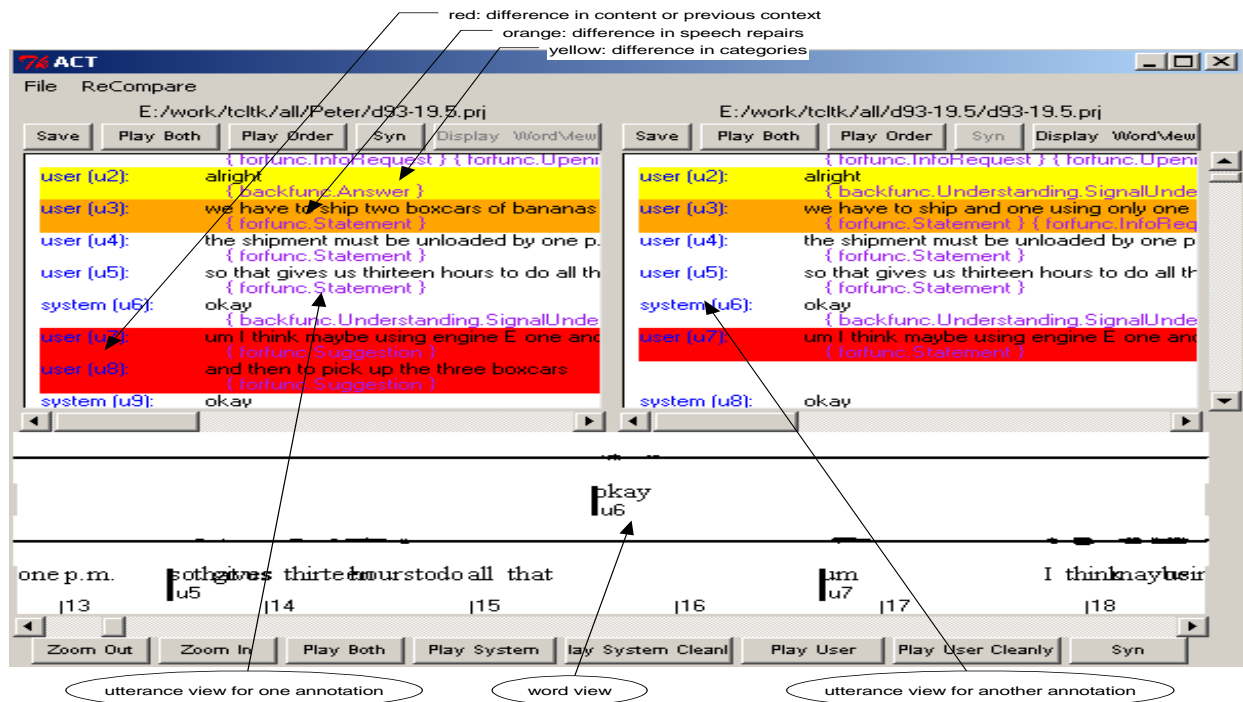


Figure 1: The ACT Interface

what is going on in the dialogue. We give a secondary view for an annotation, which we call word view [7]. It shows the speech signal, with the words, utterance boundaries (as vertical bars) and speech repairs, aligned to the signal. This view helps users understand or remember why they coded something as they did. It is ideal for viewing the exact timing of speech, especially for overlapping speech. In Examples 1 and 2 above, it is difficult to judge which segmentation is correct without referring to the word view for exact timing information. The word view from only one of the annotations being compared is displayed at a time. The user can choose which annotation to display by clicking the “Display WordView” button for that annotation. There are also “Sync” buttons in both the utterance view and the word view to synchronize which part of the dialogue is being viewed.

ACT also provides audio support that allows users to hear stretches of speech [7]. There are many different ways for the user to play the waveform. “Play Both” will play both speakers. In the word view, “Play System” or “Play User” will play the specified speaker. “Play Cleanly” will play the specified speaker without speech repairs, that is, play the intended utterances by that speaker. “Play Order” in the utterance view plays the linearization of utterances. The various play options allow the coders another means to check their annotations. For example, “Play Cleanly” should have a consistent intonation, even with the speech repairs abstracted out. If it does not, there may be something wrong with the speech repair annotation.

In addition to comparison, ACT allows users to update any of the annotations. Utterance categories annotation can be done in utterance view, while speech repairs and utterance boundaries should be done in word view. Change made in the word view are propagated into the utterance view it is related to.

4.2. Implementation

ACT is built using Tcl/Tk, packages that are included in OGI’s CSLU Speech Toolkit, and tclpp developed by Stefan Sinnige. The CSLU Speech Toolkit provides audio and wave handling. Tclpp provides object oriented extensions to Tcl/Tk. We take advantage of object oriented programming by sharing much of the code with our dialogue annotation tool, DialogueView [7].

5. CONCLUSION AND FUTURE WORK

We have described ACT, a graphical tool for comparing different annotations of a dialogue. This tool supplements existing dialogue annotation and statistical tools by providing coders a visualization that lets them focus on relevant differences. This is ideal for achieving a consensus annotation, for refining an annotation scheme, or for training novice annotators. We have also described the basis for determining which utterances have different annotations, including utterance segmentation, utterance

ent annotations, including utterance segmentation, utterance ordering, speech repairs and utterance categories.

Currently, ACT takes as input the annotation output of DialogueView [7], which includes several text files, a word file, a speech repair tag file and an utterance annotation file. We are moving toward defining a standard XML DTD as input, so that ACT can compare all annotations that are stored by this definition. By doing that, annotations by tools other than DialogueView, such as MATE [8] or Transcriber [1], can be easily compared.

Dialogue annotation involves discourse-level coding in addition to utterance-level coding. Our second improvement will be to support discourse segmentation comparison. We are now exploring discourse segment-level comparison, which turns out to be more complex than utterance-level comparison.

ACT will be available freely for non-commercial use, and distributed along with DialogueView in the CSLU Toolkit [9].

6. ACKNOWLEDGEMENTS

The authors gratefully acknowledge funding from the Intel Research Council.

7. REFERENCES

- [1] C. Barras, E. Geoffrois, Z. Wu, and M. Liberman: Transcriber: development and use of a tool for assisting speech corpora production. *Speech Communications*, 33:5-22. 2001.
- [2] J. Carletta: Accessing Agreement on Classification Tasks: The Kappa Statistic. *Computational Linguistics*, 1996.
- [3] J. Carletta, A. Isard, et, al: The Reliability of a Dialogue Structure Coding Scheme. *Computational Linguistics*, 1997.
- [4] M. Core and J. Allen: Coding Dialogs with the DAMSL Annotation Scheme. *In AAAI Fall 1997 Symposium on Communicative Action in Humans and Machines*.
- [5] P. Heeman, J. Allen: The Trains Spoken Dialog Corpus. CD-ROM, Linguistics Data Consortium, April 1995.
- [6] P. Heeman, J. Allen: Speech Repairs, Intonational Phrases and Discourse Markers: Modeling Speaker’s Utterances in Spoken Dialogue. *Computational Linguistics*, Vol. 25-4, 1999.
- [7] P. Heeman, F. Yang, S. Strayer: DialogueView: An Annotation Tool for Dialogue. To appear in the 3rd SIGdial Workshop on Discourse and Dialogue, 2002.
- [8] D. McKelvie, A. Isard, A. Mengel, et, al: The MATE workbench – an annotation tool for XML coded speech corpora. *Speech Communication*, 33:97-112. 2001.
- [9] S. Sutton, R. Cole, J. de Villiers, J. Schalkwyk, et, al: Universal speech tools: the CSLU Toolkit, *Proc. of ICSLP 1998*.