ARL-TR-8832 ● Oct 2019

Transcription Guidelines for the Army Research Laboratory (ARL) SCOUT Human–Robot Dialogue Corpus

by Claire Bonial, Cassidy Henry, Ron Artstein, and Matthew Marge

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

ARL-TR-8832 ● OCT 2019

Transcription Guidelines for the Army Research Laboratory (ARL) SCOUT Human–Robot Dialogue Corpus

by Claire Bonial and Matthew Marge
Computational and Information Sciences Directorate, CCDC Army Research Laboratory

Cassidy Henry
University of Maryland, College Park, Department of Linguistics

Ron Artstein
Institute for Creative Technologies, University of Southern California

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED (From - To)
October 2019	Technical Report	1 January 2013–30 September 2019

4. TITLE AND SUBTITLE
Transcription Guidelines for Army Research Laboratory (ARL) Human–Robot Dialogue Corpus

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Claire Bonial, Cassidy Henry, Ron Artstein, and Matthew Marge

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
CCDC Army Research Laboratory*
ATTN: FCDD-RLC-IT
Aberdeen Proving Ground, MD 21005-5067

8. PERFORMING ORGANIZATION REPORT NUMBER
ARL-TR-8832

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES
* The work outlined in this report was performed while the US Army Research Laboratory (ARL) was part of the US Army Research, Development, and Engineering Command (RDECOM). As of 31 January 2019, the organization is now part of the US Army Combat Capabilities Development Command (formerly RDECOM) and is now called CCDC Army Research Laboratory. As of February 2019, the US Army Research Laboratory has been renamed the US Army Combat Capabilities Development Command Army Research Laboratory (CCDC ARL).

ORCID ID: Claire Bonial, 0000-0002-3154-2852

14. ABSTRACT

This report provides guidelines on how to transcribe human–robot dialogue data in the context of the US Army Combat Capabilities Development Command Army Research Laboratory Situated Corpus of Understanding Transactions (SCOUT), a developmental goal of which is to facilitate the creation of a dialogue management system onboard a robot collaborating with humans in search and navigation tasks (e.g., disaster relief). This corpus was collected via a phased "Wizard-of-Oz" (WoZ) methodology, in which human experimenters perform planned dialogue and navigation capabilities of the robot during experimental trials, unbeknownst to participants interacting with the "robot". The purpose of this report is to detail the process and guidelines in which we transcribe the speech collected for the corpus during experimentation. We describe how to use the transcription software, Praat, to complete the critical task of making the experimental speech computer-readable in text format.

15. SUBJECT TERMS
speech, transcription, human-robot dialogue, natural-language processing, Praat

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Claire Bonial
Unclassified	Unclassified	Unclassified	UU	41	19b. TELEPHONE NUMBER (Include area code) (301) 394-1431

Contents

List of Figures

List of Tables

Acknowledgments

We wish to acknowledge the contributions of the following individuals for their assistance in the guidelines established in this report:

- Kimberly Pollard (US Army Combat Capabilities Development Command Army Research Laboratory [ARL])
- Clare Voss (ARL)
- Ashley Foots (ARL)
- Carla Gordon (Institute for Creative Technologies [ICT])
- Jill Boberg (ICT)
- Stephanie Lukin (ARL)
- Elia Martin (University of Maryland [UMD])
- Mitchell Abrams (Georgetown University)
- Brecken Keller (UMD)
- Sue Hill (ARL)

1. Introduction and Problem Domain

We aim to support natural-language understanding within the broader context of ongoing research to develop a human–robot dialogue system (Marge et al. 2016) to be used onboard a remotely located, autonomous agent collaborating with humans in search and navigation tasks (e.g., disaster relief). In developing this dialogue system, we are collecting and leveraging the Situated Corpus of Understanding Transactions (SCOUT), a corpus of human–robot dialogue (Lukin et al. 2018). This corpus was collected via a phased "Wizard-of-Oz" (WoZ) methodology, in which human experimenters (or "Wizards") perform planned dialogue and navigation capabilities of the robot during experimental trials, unbeknownst to participants interacting with the "robot" (Marge et al. 2017). The purpose of this report is to detail the process and guidelines by which we transcribe the speech collected for the corpus.

The WoZ method is bottom-up in the sense that we do not assume that we can know a priori how humans communicate with a robot in a shared task. Instead, our WoZ methodology facilitates a data-driven understanding of how people talk to robots in our collaborative domain. Similar to DeVault et al. (2014), we use the WoZ methodology only in the early stages of a multistage development process to refine and evaluate the domain, and provide training data for automated dialogue system components. In all stages of this process, participants communicating with the "robot" speak freely, even as increasing levels of automation are introduced in each subsequent stage or "experiment;" the iterative automation process utilizes previous experiments' data. Currently, we are in the fourth experiment of the ongoing series, with transcription completed as described in this report for experiments 1–3.

During a data collection experimental trial, a naïve participant (unaware of the wizards) performs collaborative exploration with a remotely located robot. The participant instructs a robot to navigate through an unfamiliar house-like environment, asking it to find and count objects of interest in a search task (e.g., hard-to-find objects such as shoes and shovels). In reality, the participant is not speaking directly to a robot, but to an unseen experimenter, the Dialogue Manager (DM) Wizard, who listens to the participant's spoken instructions, and responds with text messages in a chat window or passes a simplified text version of the instructions to another experimenter, the Robot Navigator (RN) Wizard, who joysticks the robot to complete the instructions. Our experimental setup is presented in Fig. 1. Even though human experimenters control autonomous behaviors, the physical robot exists and sends information back to the participant in the form of natural-language dialogue and task-relevant photographs. Given that the DM acts

as an intermediary, passing communications between the participant and the RN, the dialogue takes place across multiple conversational floors. The flow of dialogue from participant to DM, DM to N, and subsequent feedback to the participant can be seen in Table 1.

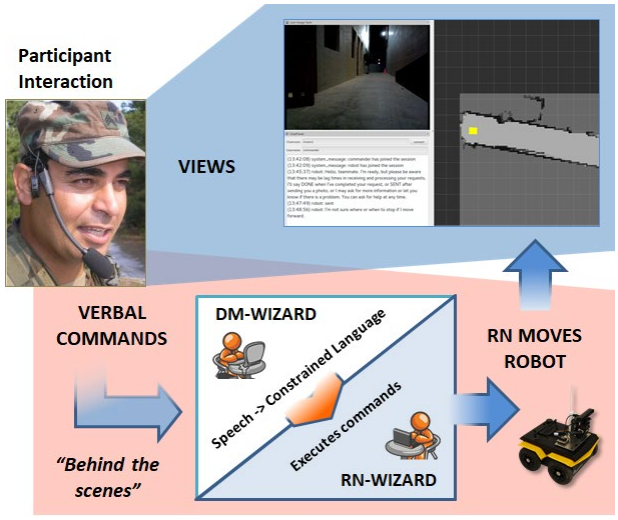

Fig. 1 WoZ setup for our experiment: participant instructs robot with two experimenters standing in for robot capabilities

Table 1 Example of the flow of dialogue from participant to experimenters—a minimal transaction unit (TU) in existing SCOUT dialogue annotation, which contains an instruction initiated by the participant, its translation to a simplified form (DM to RN), and the execution of the instruction and acknowledgement of such. TU, Antecedent (Ant), and Relation (Rel) type annotations are indicated in the right columns and described elsewhere (Traum et al. 2018).

No.	Left floor		Right floor		Annotations		
	Participant	DM → Participant	DM → RN	RN	TU	Ant	Rel
1	Move forward 3 ft	…	…	…	1		…
2	…	ok	…	…	1	1	ack-wilco
3	…	…	Move forward 3 ft	…	1	1	trans-r
4	…	…	…	Done	1	3	ack-done
5	…	I moved forward 3 ft	…	…	1	4	trans-l

The transcription guidelines detailed in this report facilitate the critical step of taking speech data collected in human–robot dialogue experimentation and making it computer-readable text to be then annotated, as seen in Table 1, and leveraged in other ways as training data for a dialogue management system.

2. Praat Transcription Tool

For transcribing speech data, we selected Praat (Boersma et al. 2019). Praat is a powerful software tool used commonly among linguists for labeling and transcribing sound. Praat is available for free at http://www.fon.hum.uva.nl/praat/.

To download, select the version appropriate for your machine (32 or 64 bit). Extract the executable file to an appropriate place that is easy to find or create a shortcut—no other installation is necessary. The screenshots and shortcuts shown are for Windows, but the software is available for other operating systems as well.

3. Transcribing with Praat

3.1 Opening a Sound File in Praat

After installing and opening Praat, the first step is to open an audio file in the program (Fig. 2).

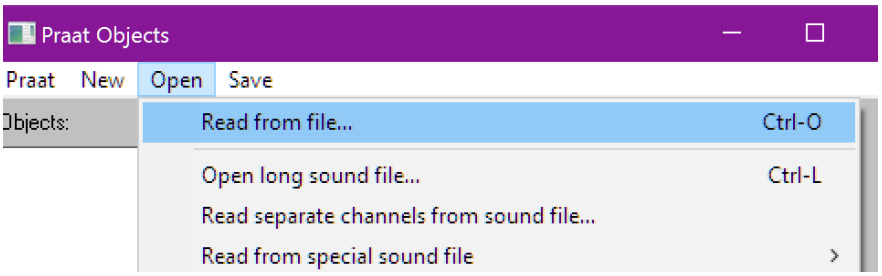

Fig. 2 Open the sound file in Praat: Open -> Read from file… or command shortcut Ctrl-O

After completion, the screen should show the audio file selected through the file explorer in the Praat window (Fig. 3).

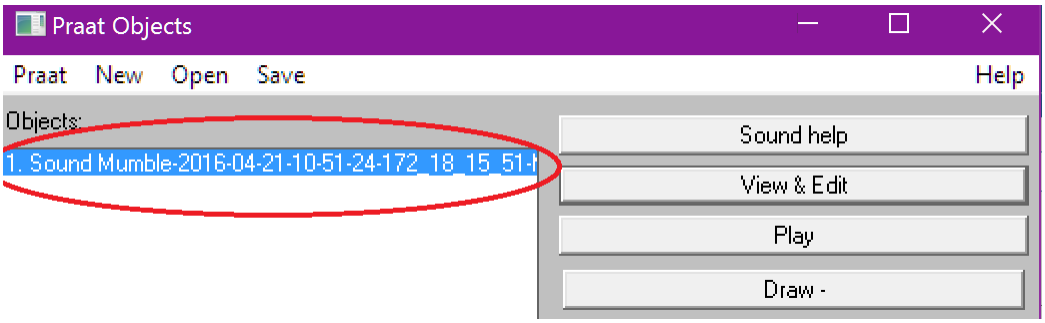

Fig. 3 Select the audio file

Press “View & Edit” on the right to get a view of the audio file with its spectrograms or simply play the audio file. As we want to transcribe the contents of the file in a time-aligned manner, there are additional steps prior to transcription: we will be adding space in which the content of the audio files can be transcribed in real time.

3.2 Adding Tiers

The space upon which transcription and annotation in Praat is done is called a “tier”. This is because, in the graphical user interface (GUI) view of the audio file and annotation/transcription space, the information is displayed in stacked tiers. One can have any number of tiers needed and can name them to inform what they contain. The tiers are formatted as TextGrid files, which is a file format that is both human and machine-readable in any text editor.

To add a tier, click the “Annotate” button, selecting the “To TextGrid (silences)…” option (Fig. 4). The reason we use the silences tier is that, with the right configuration, this process will automatically segment our tier for each instance in which the microphone was actively being spoken into as opposed to silence, which greatly expedites our process and reduces the amount of labor required to transcribe audio.

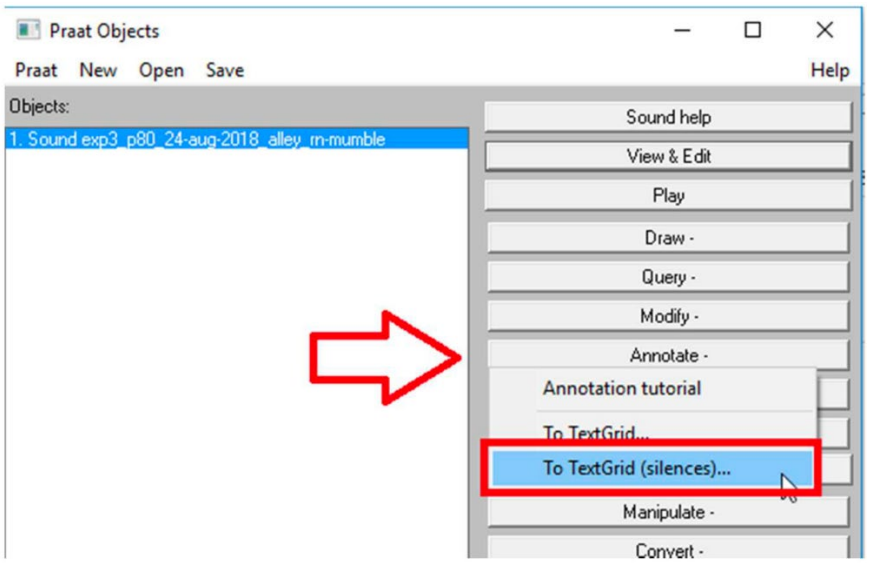

Fig. 4 Add a silences tier to TextGrid

Figure 5 shows the values we generally work with for this feature. Make sure to delete all text from the “Sounding interval label” box and use “<SIL>” for silent interval labeling. This generates a TextGrid that is blank in the spaces where transcription will occur, and “<SIL>” marks silent intervals that can be ignored due to the lack of any audio content.

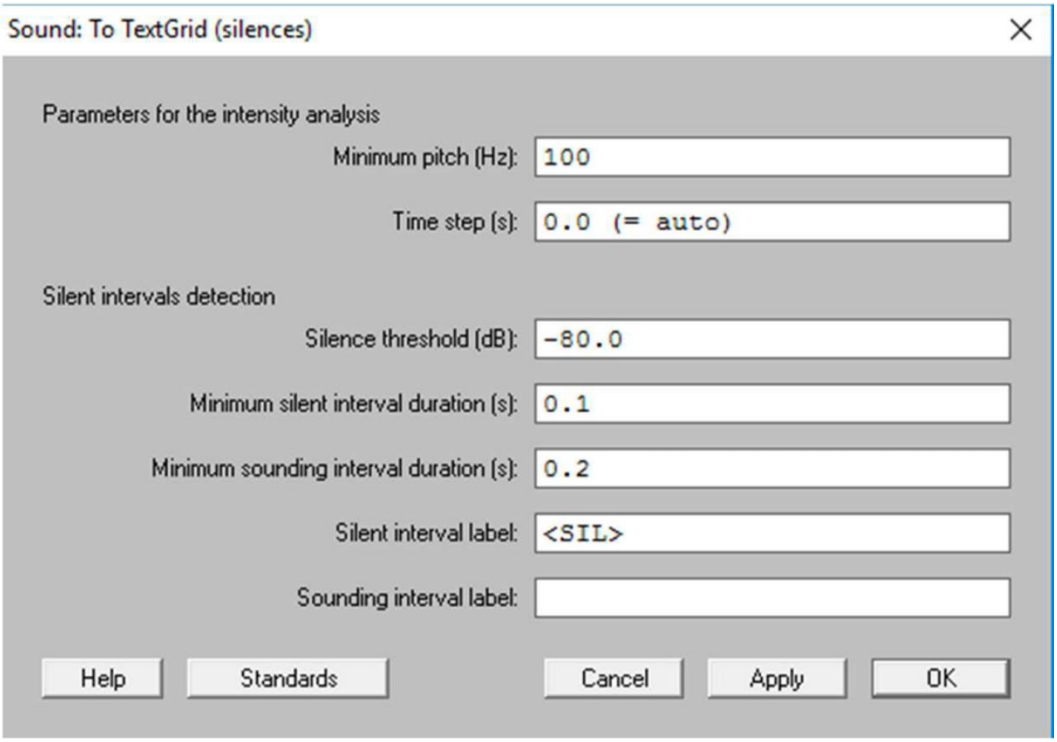

Fig. 5 Settings for silence thresholds

Hit "OK" or "Apply". This will create a TextGrid file that appears as a new Praat "Object".

3.3 Beginning the Transcription

Select both files (Ctrl+click), then click "View and Edit" (Fig. 6).

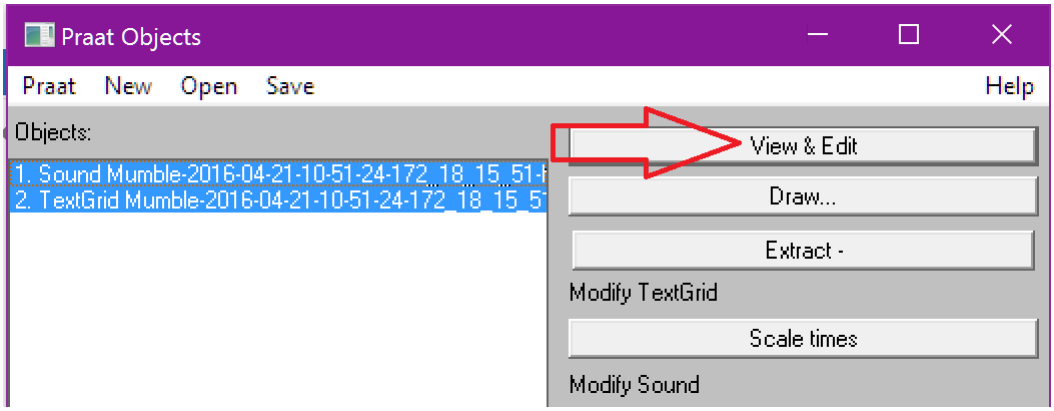

Fig. 6 View and edit the sound and TextGrid file

Now, a Praat window will open, containing a waveform of the audio file, a spectrogram (only shows at 10 s and below by default), and the tiers from the TextGrid. It should look like Fig. 7.

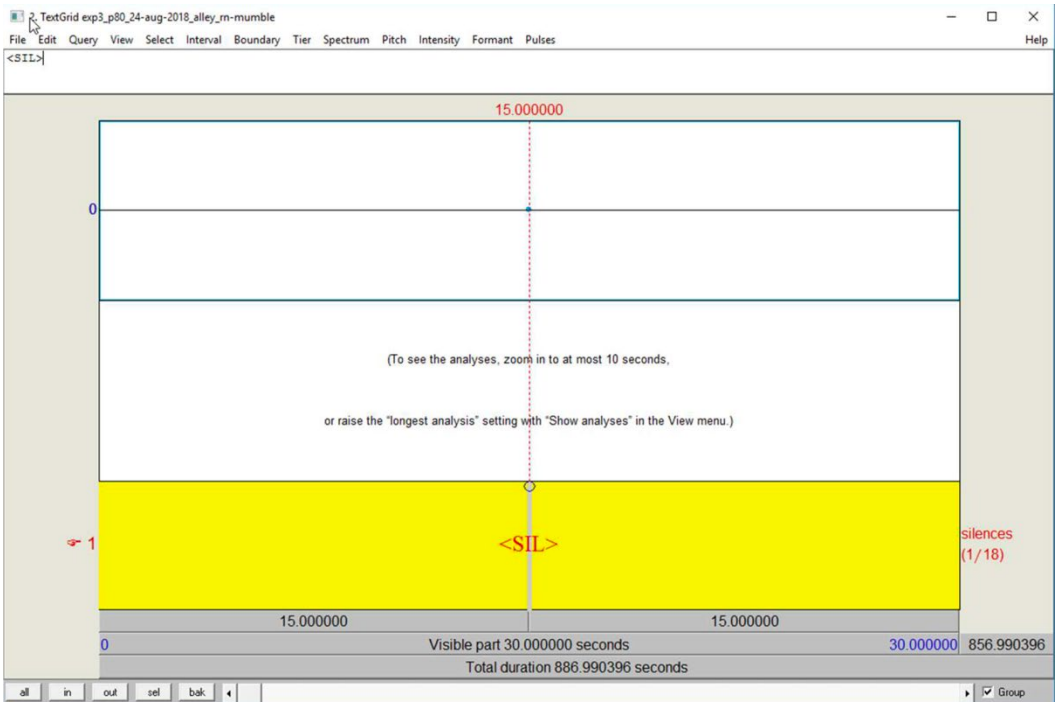

Fig. 7 Praat default view

If the view/edit window has opened, the audio has successfully opened, even if there is a flat-appearing waveform on display. Praat defaults to a zoomed-in, 30-s portion of the audio. Click "all" in the lower-left hand corner to display the full audio file (Fig. 8).

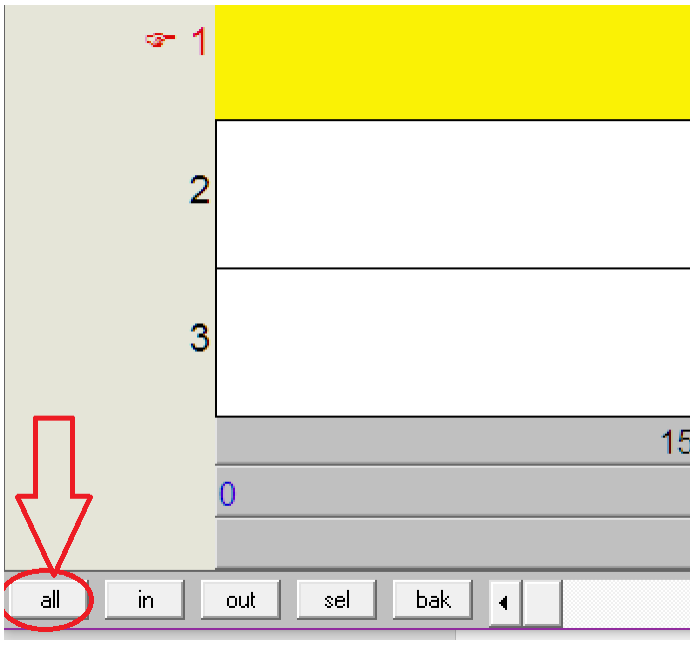

Fig. 8 Zoom-out view by selecting "all"

The other buttons are as follows: *in*, zoom in on the waveform; *out*, zoom out on the waveform; *sel*, only show the selected area of the waveform (shows up in pink on waveform); and *bak*, go back from a selection. After clicking "all," the windows should show the full waveform (Fig. 9).

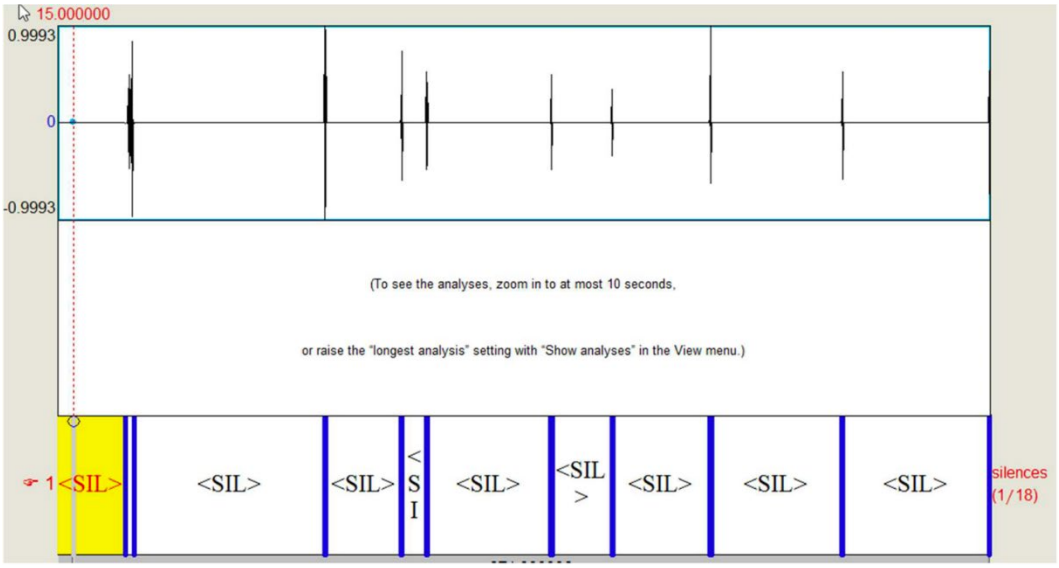

Fig. 9 Full waveform view

Sometimes the process of automatically segmenting the utterance boundaries is overzealous (in the sense that it automatically segments out very quiet noises that are often not speech), but it is important to keep the threshold for what is considered "silence" low so that we do not accidentally exclude any quiet utterances. If there are additional segments that should be part of a larger section of ongoing silence as a result of the automatic segmentation, one can delete the boundaries of that segment. Deleting boundaries is not difficult in Praat and is described in more detail in the following paragraphs.

When additional segments are needed (see Section 4 for segmentation practices), one can also create new boundaries for the transcriptions by holding down the left-click button and highlighting a waveform spike (see Section 5 for reading the waveforms/spectrograms), then pressing "Enter" to surround the utterance with two boundaries (we suggest zooming in to do this for more precise boundaries). Individual boundaries can be made by clicking on the appropriate point in the waveform and hitting "Enter". For a single boundary, one can simply left-click on the location in the waveform/spectrogram where the boundary should go, then hit "Enter". Alternatively, one can also add boundaries within existing boundaries by clicking on and, if needed, dragging over the area to be bounded, then clicking on "Boundary" in the top menu "Add on selected tier".

To adjust boundaries, click on the bounded area so that it is highlighted, then click on the blue boundary to adjust and drag it where it should be using the mouse.

To delete boundaries, click on the bounded area (or individual boundary, if only one needs removal) to delete so that it is highlighted, then click on the blue boundary. Once the boundary is selected, one can either use the keyboard shortcut (Alt+Backspace) or click “Boundary” in the top menu, then select “Remove”. This will remove the boundary and concatenate the previously bounded area with the next bounded area. It will also concatenate any annotations or “<SIL>” markers, so be sure to delete or adjust transcriptions as needed.

Figure 10 shows an utterance selected, with all the periods of silence marked as “<SIL>”.

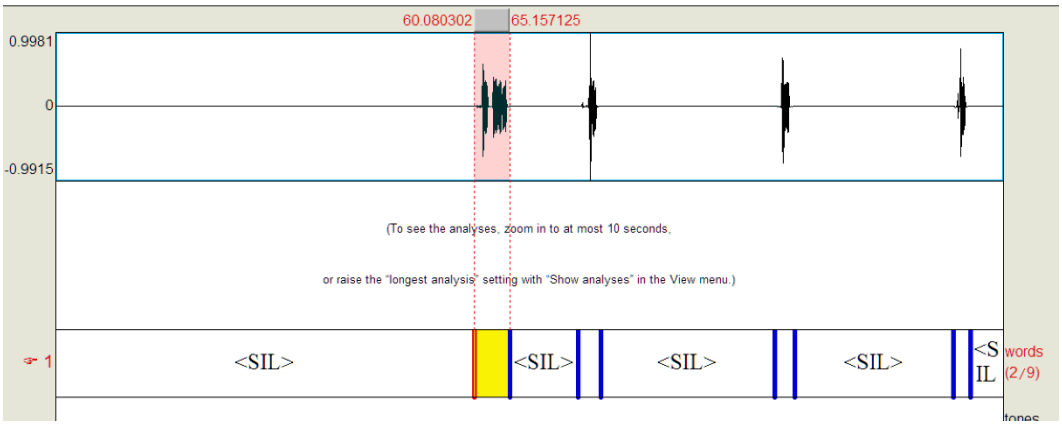

Fig. 10 Selected utterance

To play the audio, click the selection and either hit “Tab” or click the segmented space at the bottom of the interface (Fig. 11).

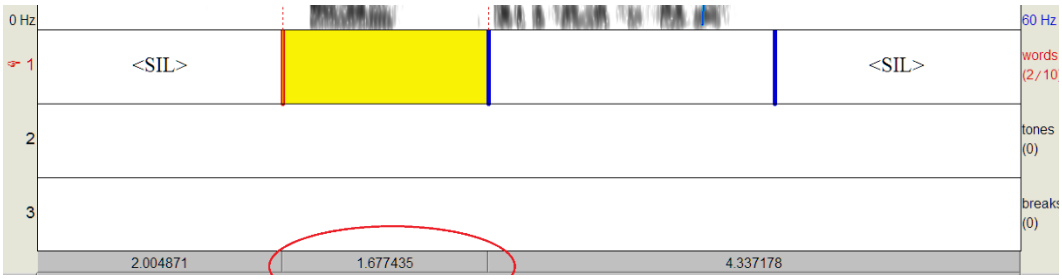

Fig. 11 Click the circled space to play the selection only

Then, segment as needed per utterance as conventions describe in the next section. One can type the transcription contents when the segment is selected or click and type into the text box at the top of the interface (Fig. 12).

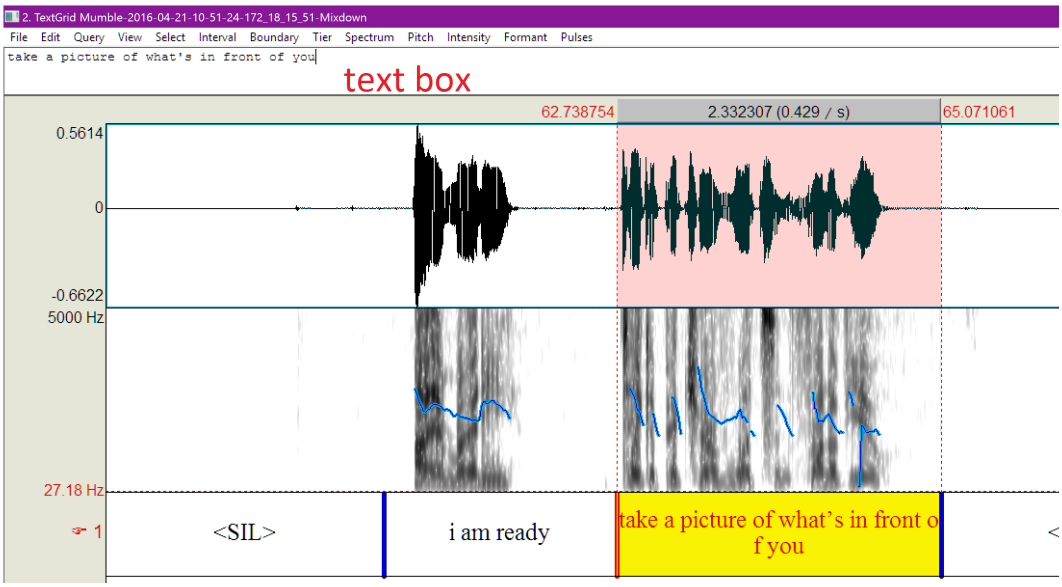

Fig. 12 Type the transcription of the selected segment into the text box at the top of the window

Be sure to *save early and often*. Praat does not efficiently save large chunks of information at a time. To save, one can use the standard Ctrl+S keyboard shortcut, or in the menu of the view/edit window of Praat (shown above), click on "File". then "Save TextGrid as text file". This creates a "*.TextGrid" file.

For the SCOUT corpus development pipeline, the TextGrid file should be named according to the conventions of the original sound file (e.g., save exp3_p61_24-sep-2018_alley_cmd-mumble.wav as exp3_p61_24-sep-2018_alley_cmd-textgrid .TextGrid). When finished with all transcription for a file, copy and commit the TextGrid file to the appropriate folder on the SCOUT corpus subversion system (SVN) (e.g., svn_botlanguage/Data/Experiment3/structured_data/cmd-textgrid or rn-textgrid, respectively).

4. Project-Specific Transcription and Segmentation Practices

All transcription conventions for SCOUT follow Ron Artstein's transcription conventions (see Appendix A), with few modifications and the addition of segmentation.

Segmentation:

- Utterances are segmented into boundaries on first and foremost on a per-command basis. For instance, the utterance "move forward 5 feet and turn right then take a picture" would be bound into three separate interval selections in Praat from the original single interval:

move forward 5 feet | *and turn to the right* | *then take a picture*

- Exception to segmentation: RN says "done and sent"—this is one segment.
- <SIL> marks periods of silence between utterances. This is automatically added by Praat for "silence" below a certain threshold of audio for a duration of 0.1 s or more. One may need to delete <SIL> segments that have been automatically added but are less than 1 s and interrupt a single command (see next bullet).
- A semantically continuous command may include silent (or nearly silent) pauses (e.g., "Turn right … 45 degrees"):
    - If the pause is between 0.2 and about 1 s, and one can hear some background sound (indicating the participant is holding down the push-to-talk button and therefore probably intends to continue speaking), it is likely that this should be treated as one segment with intervening pause notation indicating the duration of the pause: *turn right <pause, .24> 45 degrees*. The pause duration measurement is easily obtained in Praat by highlighting over the pause segment, which will cause the duration to appear within the highlighted area.
    - If the pause is greater than about 1 s and one cannot hear any background sound (indicating the participant has released the push-to-talk button), it is likely that this should be treated as two segments even though it could be understood semantically and syntactically as one command: *turn right* | *45 degrees*.

Other project-specific transcription conventions:

- There is a clap noise used for syncing at the beginning of each experimental trial in experiment 3 onward. Transcribe this clap as <loud noise>.
- Experimenter speech that can be heard in the commander audio file should be transcribed as <X: *words*>.
- Occasionally, one will hear a musical tone that plays on the commander's computer to notify them that they have an important message from the DM; transcribe these tones as <notification sound>.
- <disfl> is a tag used to mark speech disfluencies in place (e.g., rotate forty five grees<disfl> degrees to the right)
- <no speech> is used when some background noise can be heard but no speech is occurring because the push-to-talk button is held down for significant amount of time (about 1 s or greater) prior to speech or after.

The relevant portion of Dr Artstein's guidelines are copied in Appendix A of this report, detailing all other transcription conventions.

5. Reading Spectograms for Precise Segmentation

A natural question that arises during segmentation and subsequent transcription is where, precisely, to place boundaries. Spectrograms and waveforms visible in Praat are useful tools for precise boundary marking. With some training, it is easy to identify speech sounds and how they appear in their various visualizations.

Generally, a boundary can be placed once significant activity is visible on waveforms and/or spectrograms. In the case of waveforms, acoustic activity is visible when there is a disturbance in the line, and for most cases, this is an easy way to determine when to begin your boundary if it needs to be precise. Figure 13 shows a view in Praat of a zoomed in slice of the word "take," specifically on the first consonant /t/.

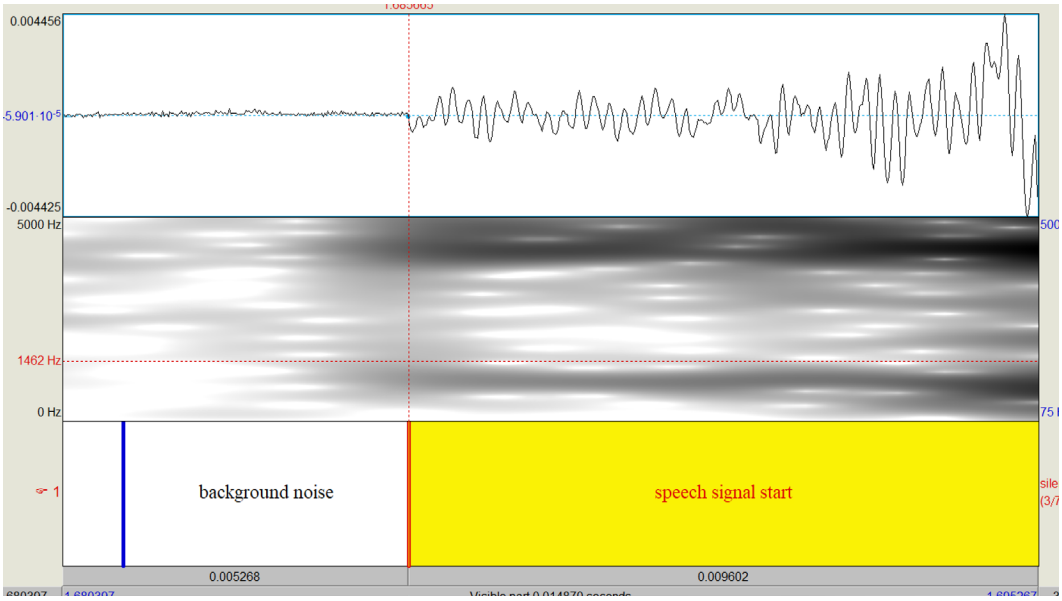

Fig. 13 A screenshot of Praat showing the onset of the consonant /t/, illustrating the difference in the waveform between background noise and the onset of a consonant

The first boundary showing light activity in the waveform is just background noise, which can be caused by a number of things: a computer fan, air conditioning—anything that could cause a disturbance in the air. The second boundary covers the moment the /t/ begins to be produced. Vowels and other voiced sounds are easiest to spot in a spectrogram in that they exhibit periodicity (repeating patterns) in their waveforms, which correspond to harmonic frequencies visible in the spectrogram, but consonants can be separated from background noise by comparing if a pattern

seems marked relative to the background noise. In Fig. 13, it is noticeably more active than the background noise.

Some cases to be mindful of when placing precise boundaries for SCOUT data are when plosive (stop) consonants are at the boundary, as they appear to be silent on the spectrogram when they are the initial consonant on a boundary, especially when devoiced. Stop consonants in English are as follows: [voiced: b, d, g; devoiced: p, t, k], so any words that begin with these sounds at a boundary are ones to be mindful of. In Fig. 14, there is an example of /d/ in the word "don't," where the portion of the consonant that represents closure (highlighted on the waveform in pink) has minimal waveform activity (but different from the silence that briefly precedes it). The closure could be misinterpreted by someone who does not know how to read waveforms and spectrograms as background noise, when it is in fact the onset of speech. The portion immediately following the highlighted section is the release burst of the stop, which is where some people would erroneously label the beginning of the sound (as it becomes most audible at this point).

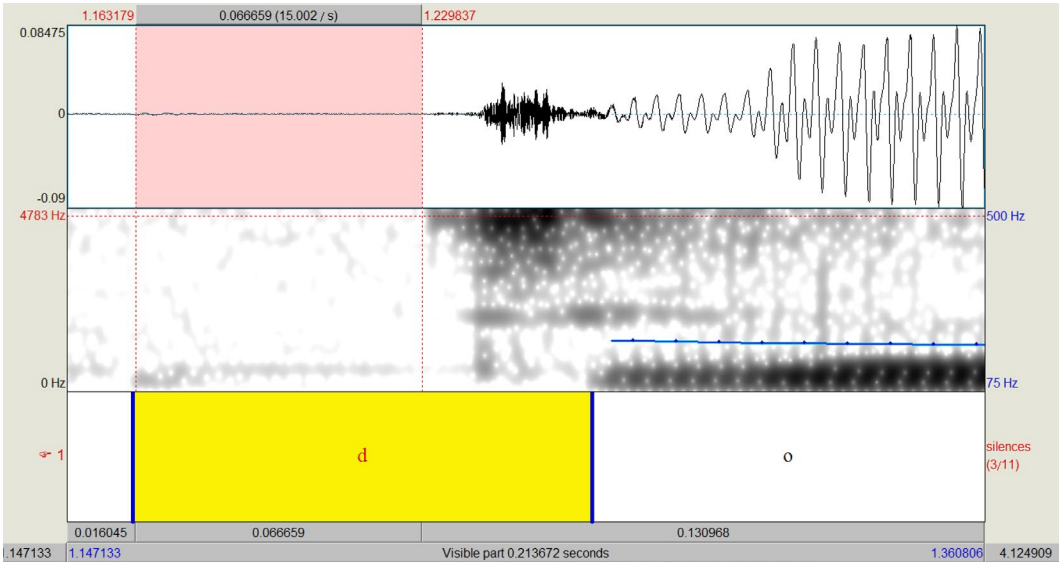

Fig. 14 A screenshot of Praat showing the onset of /d/, illustrating the fact that closure could be misinterpreted as background noise

For more detailed information on reading spectrograms and acoustic phonetics in general, refer to introductory acoustic phonetics texts such as Keith Johnson's 2011 *Acoustic and Auditory Phonetics* book.

6. Postprocessing: Creating a Transcript from a TextGrid File

Once the transcription is complete in Praat, it is necessary to create a text transcript of the work. The TextGrid itself is a specially formatted file openable in any text editor, but there is a Praat script that one can copy and use to extract a timed transcript:

- svn_botlanguage/Software/Praatscripts/save_conversation_tiers_as_text_file.praat

The content of this script can also be found in and copied from Appendix B, and is available for download online as part of The Speech Corpus Toolkit for Praat (SpeCT) (Lennes 2017).

Save the script in an easily accessible location, as it will be needed to open the file in Praat.

To open the script, go to the "Praat Objects" window, then select Praat->Open Praat Script… (Fig. 15).

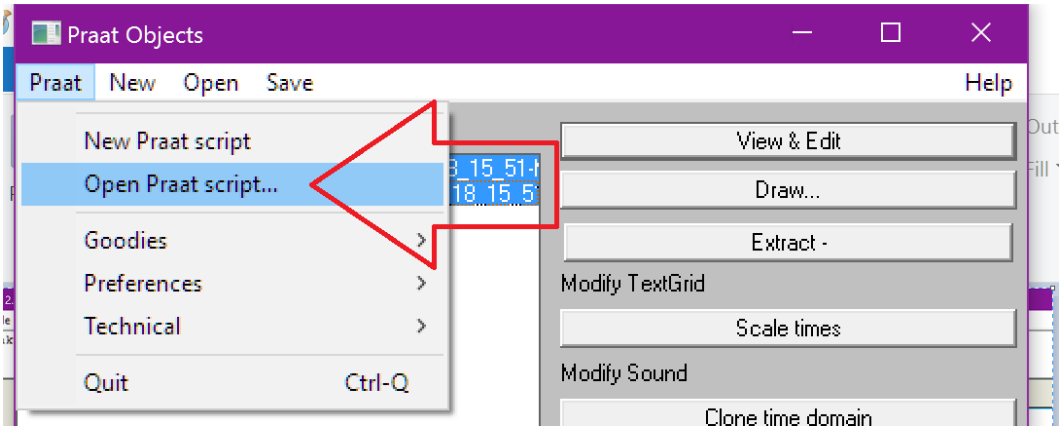

Fig. 15 Open the Praat script

Now, locate and open the Praat script for creating a transcript (Fig. 16).

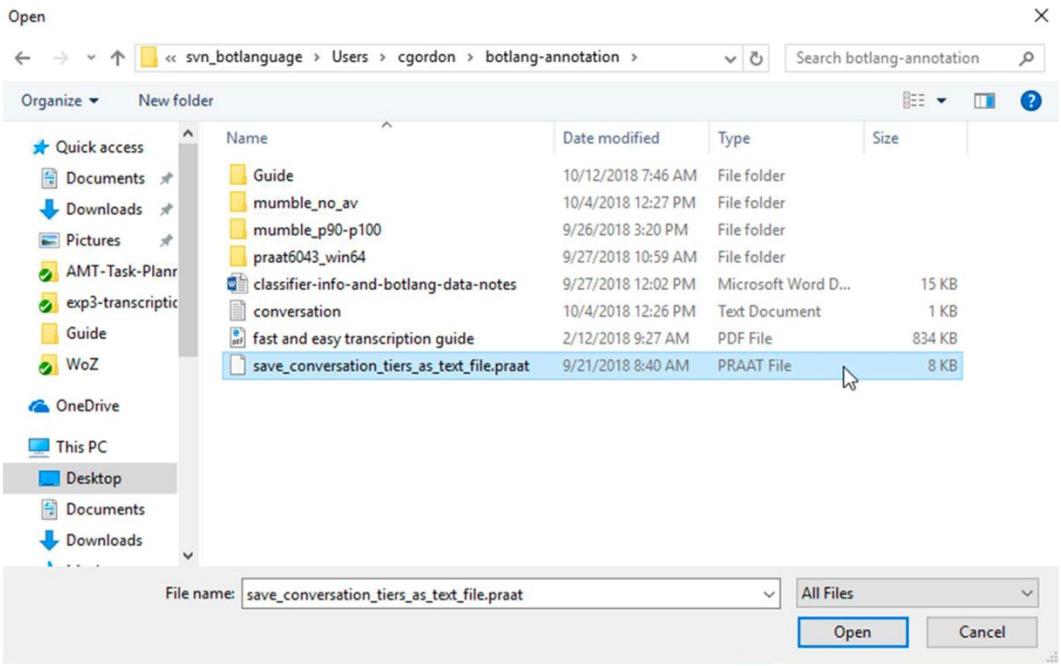

Fig. 16 Select the Praat script

Once opened, it should look like Fig. 17.

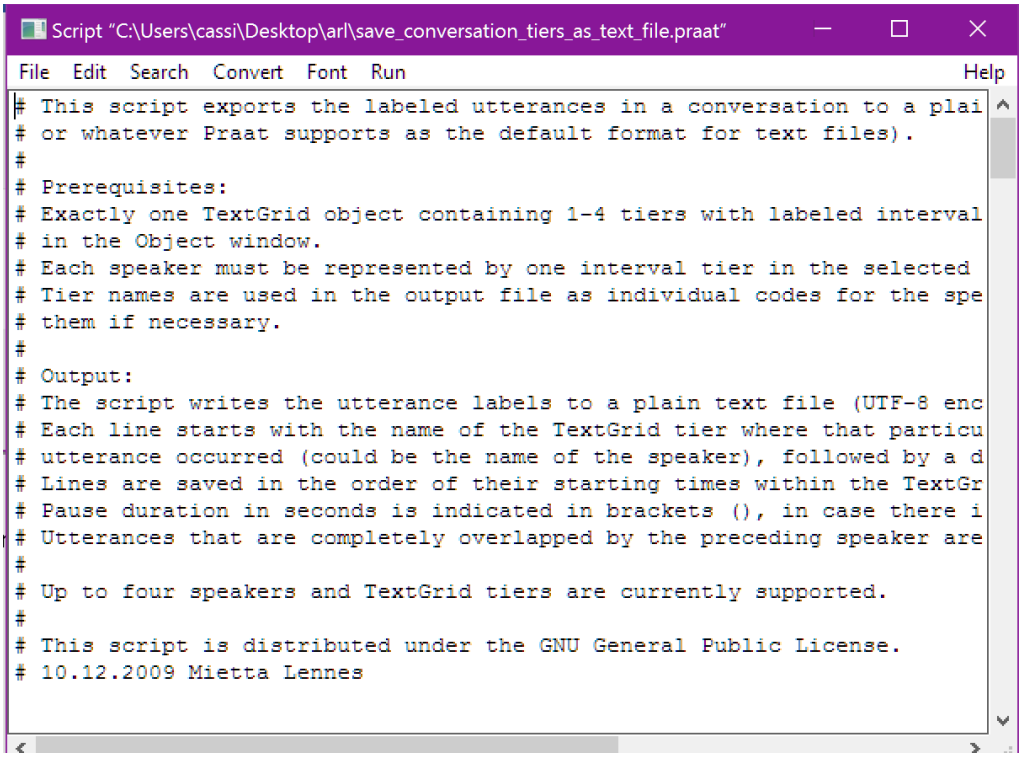

Fig. 17 View of the Praat script

To run the script, make sure to only have the TextGrid file selected—one cannot extract text from audio directly (otherwise this guide would be null and void). An error message will display if one does not select only one TextGrid. In the previous window, click Run->Run (Fig. 18).

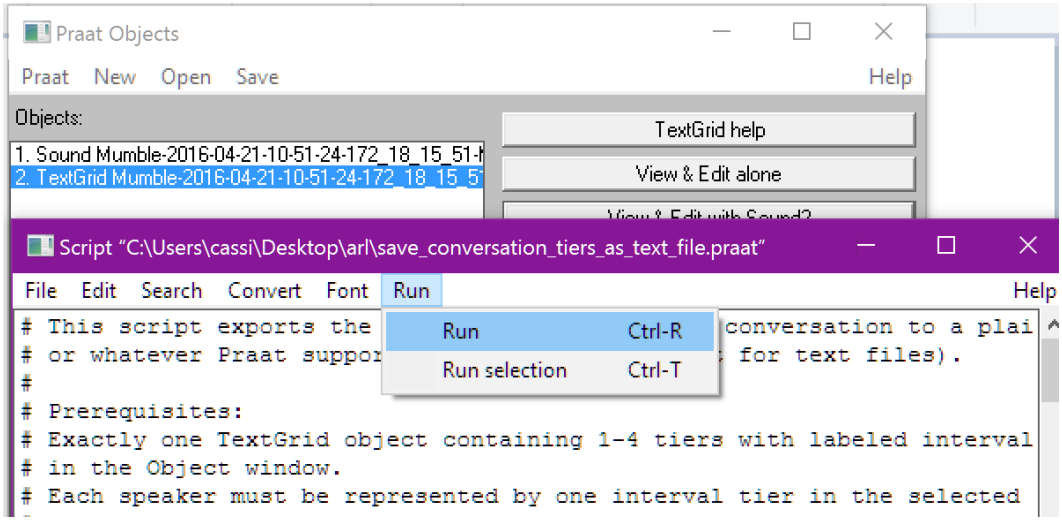

Fig. 18 Run the Praat script

A window will pop up prompting for tier names (enter the tiers to be extracted to text) and other options such as "filename". Maintain the file naming conventions of the original audio file, but add "transcript" (example in screenshot in Fig. 19): exp3_p95_06-aug-2018_house2_cmd-transcript.txt. Ensure that only "Insert start times" is selected below the tier names, as shown.

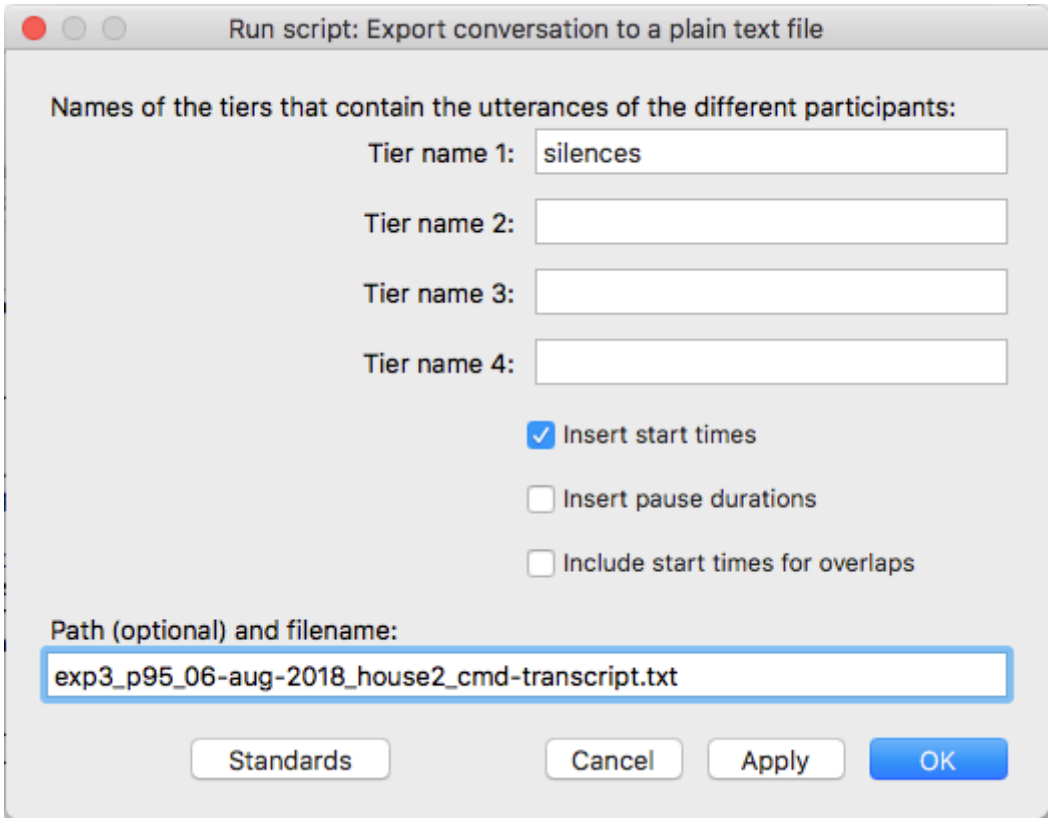

Fig. 19 Name tiers

Hit "OK" and the script will process (Fig. 20).

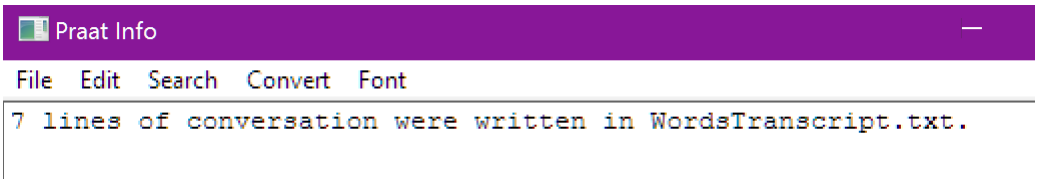

Fig. 20 Praat script processing message

It will save the text file in the same location as the script, which is the other reason why we want it in an easy-to-locate place (Fig. 21).

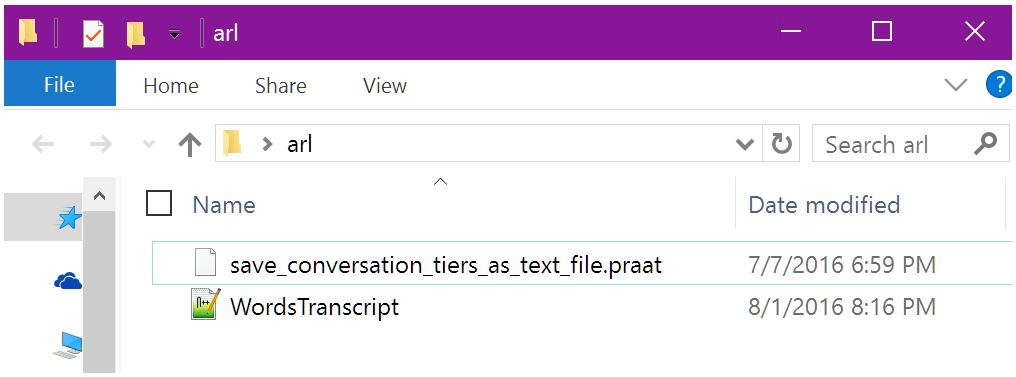

Fig. 21 Output transcript file appears

Figure 22 shows what the output looks like.

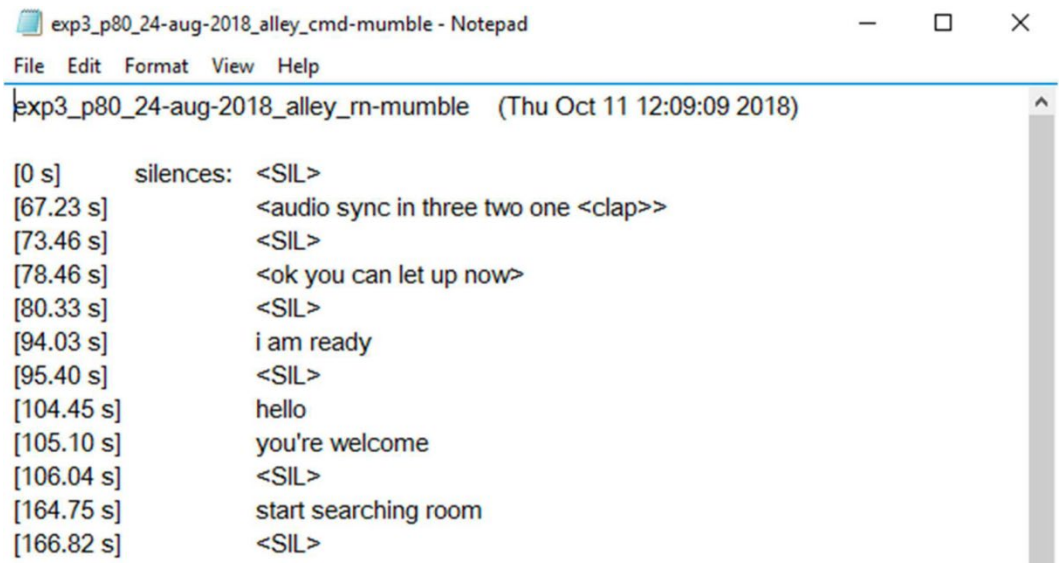

Fig. 22 Sample text transcript

The basic structure of the file looks like this:

- [FILENAME] (Date saved)
- [0 s] TIER NAME: contents
- [x.x s] contents

Copy and commit the .txt file to the SVN in the appropriate folder (e.g., svn_botlanguage/Data/Experiment3/structured_data/cmd-transcript or rn-transcript, respectively).

Once both the TextGrid and transcript files are committed to the SVN, transcription is complete.

7. Conclusion

In this report, we have outlined procedures for transcribing human–robot dialogue speech data. These guidelines have been under development and used throughout experiments 1–3. In experiment 4, we have further automated the DM role and plan to use Google Automatic Speech Recognition, which provides automatic transcription. Although this transcription is not of the same quality as manual transcriptions, we will explore using this as a baseline system followed by manual corrections.

8. References

Boersma P, Weenink D. Praat: doing phonetics by computer [Computer program]. Version 6.1.01. Amsterdam (The Netherlands): Phonetic Sciences, University of Amsterdam; 2019 [accessed 2019 Aug 14]. http://www.praat.org/.

DeVault D, Artstein R, Benn G, Dey T, Fast E, Gainer A, Georgila K, Gratch J, Hartholt A, Lhommet M, et al. SimSensei Kiosk: a virtual human interviewer for healthcare decision support. In: Proceedings of the 2014 International Conference on Autonomous Agents and Multi-Agent Systems; 2014 May. p. 1061–1068. International Foundation for Autonomous Agents and Multiagent Systems.

Johnson K. Acoustic and auditory phonetics. Howard DM, editor. New York (NY): Wiley-Blackwell; 2011 Aug 15.

Lennes M. SpeCT - Speech Corpus Toolkit for Praat (v1.0.0). First release on GitHub, Version 1.0.0. Zenodo; 2017 Mar 8. doi: http://doi.org/10.5281/zenodo.375923.

Lukin SM, Gervits F, Hayes CJ, Leuski A, Moolchandani P, Rogers III JG, Traum D. Scoutbot: a dialogue system for collaborative navigation. Ithaca (NY): Cornell University; 2018. http://arXiv:1807.08074.

Marge M, Bonial C, Byrne B, Cassidy T, Evans AW, Hill SG, Voss C. Applying the Wizard-of-Oz technique to multimodal human-robot dialogue. In: Proceedings of the IEEE International Conference on Robot and Human Interactive Communication (RO-MAN) 2016; 2016.

Marge M, Bonial C, Foots A, Hayes C, Henry C, Pollard K, Artstein R, Voss C, Traum, D. Exploring variation of natural human commands to a robot in a collaborative navigation task. In: Proceedings of the First Workshop on Language Grounding for Robotics; 2017 Aug. p. 58–66.

Traum D, Henry C, Lukin S, Artstein R, Gervits F, Pollard K, Bonial C, Lei S, Voss C, Marge M, Hayes C, Hill S. Dialogue structure annotation for multi-floor interaction. In: Proceedings of the 11th International Conference on Language Resources and Evaluation (LREC-2018); 2018 May.

Appendix A. Institute for Creative Technologies (ICT) Transcription Practices by Dr Ron Artstein

This appendix appears in its original form, without editorial change.

The following is an excerpt of the relevant portion of transcription guidelines developed by Dr Ron Artstein at the Institute for Creative Technologies of the University of Southern California.[1]

A.1 Orthography

1. Words are transcribed using standard American English spelling conventions, except as noted below.
2. Transcriptions use only lowercase letters.
3. All words are spelled out without abbreviation: *doctor perez*, not *dr. perez*.
4. Letter names are transcribed as a single letter separated by spaces: *a b*.
5. Numbers:
    a) All numbers are spelled out: *one, fifteen, twenty five*.
    b) Numbers are transcribed as pronounced: *one hundred and first* is distinct from *one o first*.
    c) We do not use digits for transcriptions.
6. Acronyms: ALL acronyms are spelled out as they are in standard English (ARL, FBI, TV); this includes acronyms which are pronounced letter by letter and acronyms pronounced as a word.
7. Military call signs are transcribed as pronounced: *alpha*.
8. Contractions are transcribed in the conventional way: *we'll, can't*. This also applies to non-standard contractions which have a conventional spelling: *ain't, wanna, 'cause.*
    - The apostrophe character for contractions in English is the ASCII character (straight quote, U+0027), not any "smart" or curly character. The reason is that some of the software we use is not aware of character encodings or multibyte characters.
9. Minor pronunciation deviations from standard are ignored: *living, not livin'*.
    - The purpose of this convention is to train the speech models to associate such deviant pronunciations with the intended word.
10. Generally, words are transcribed using unaccented Latin-script letters only.

[1] Artstein R. Institute for Creative Technologies, University of Southern California, Los Angeles, CA. Personal communication, 2018.

a) Accented characters in English words are converted to their unaccented counterparts: *cafe*, not *café*.

b) Words in foreign languages that use the Latin script are transcribed using native language conventions, except that accented characters are converted into their ASCII counterparts: *como estas*, not *cómo estás*.

c) Words in foreign languages that do not use the Latin script are transcribed using an accepted transliteration into unaccented Latin-script letters.

- The reason for the above conventions is that some of the software we use is not aware of character encodings or multibyte characters, and we have no general way of indicating the character encoding of transcription text files. Specific projects may deviate from the above conventions – for example, Pashto in the CHAOS project is transcribed using native Pashto (Arabic-script) characters. In such cases a character encoding should be agreed upon; the recommended encoding is UTF-8.

11. Generally, we do not use punctuation in our transcriptions.

A.2 Disfluencies

12. All repetitions and hesitations are transcribed as pronounced: *i i i think, eh, um*.

13. If a person misspeaks (e.g. says "happen" instead of "happened"), write the word actually pronounced. The reason for this convention is that such problems are better handled by the NLU component, so we want to train the ASR to give the actual word pronounced and to train the NLU to handle such input.

14. If speech is cut-off, put down the complete intended word, followed by a comment with the part that was actually pronounced in angle brackets: *people* <*peop*>. The comment is just for human readers; the reason for transcribing a whole word is to avoid confusing the processing modules by training them on non-words.

15. Unrecognizable words are written as *xxx*.

A.3 Non-speech

16. When a speaker produces an audible non-speech sound, put it down as a comment between angle brackets: <cough>, <laughter>, <sigh>.

17. Other comments also go between angled brackets (for example speaker identification in short audio files). The precise placement (at the beginning or end of a line) varies by project.

Appendix B. Additional Praat Resources

This appendix appears in its original form, without editorial change.

Keyboard shortcuts: http://www.fon.hum.uva.nl/praat/manual/Keyboard_shortcuts.html

FAQ: http://www.fon.hum.uva.nl/praat/manual/FAQ__Frequently_Asked_Questions_.html

Intro tutorial: http://www.fon.hum.uva.nl/praat/manual/Intro.html

Tons of info on its website: http://www.fon.hum.uva.nl/praat/

Other beginners' tutorials: http://www.fon.hum.uva.nl/praat/manualsByOthers.html

The following is Save_conversion_tiers_as_text_file.praat script, which can also be downloaded from the SpeCT website, https://lennes.github.io/spect/:

```
# This script exports the labeled utterances in a conversation to a plain text file (UTF-8,
# or whatever Praat supports as the default format for text files).
#
# Prerequisites:
# Exactly one TextGrid object containing 1-4 tiers with labeled intervals must be selected
# in the Object window.
# Each speaker must be represented by one interval tier in the selected TextGrid object.
# Tier names are used in the output file as individual codes for the speakers, so you should change
# them if necessary.
#
# Output:
# The script writes the utterance labels to a plain text file (UTF-8 encoding), one utterance per line.
# Each line starts with the name of the TextGrid tier where that particular
# utterance occurred (could be the name of the speaker), followed by a double colon : and a tab.
# Lines are saved in the order of their starting times within the TextGrid object.
# Pause duration in seconds is indicated in brackets (), in case there is no overlap.
# Utterances that are completely overlapped by the preceding speaker are marked in square brackets [].
#
# Up to four speakers and TextGrid tiers are currently supported.
#
```

```
# This script is distributed under the GNU General Public License.
# 10.12.2009 Mietta Lennes


# Ask the user for some details:
form Export conversation to a plain text file
comment Names of the tiers that contain the utterances of the different
participants:
sentence Tier_name_1 F1
sentence Tier_name_2 F2
sentence Tier_name_3
sentence Tier_name_4
boolean Insert_start_times yes
boolean Insert_pause_durations yes
boolean Include_start_times_for_overlaps yes
comment Path (optional) and filename:
text Filepath conversation.txt
endform

gridname$ = selected$ ("TextGrid", 1)
# Convert the special characters in the TextGrid into Unicode format
#Nativize
total_duration = Get total duration
start_time = Get start time
# Check whether the user wishes to overwrite an existing file by the same name:
if fileReadable(filepath$)
	pause File 'filepath$' exists! Delete it and continue?
	filedelete 'filepath$'
endif
date$ = date$()
fileappend 'filepath$' 'gridname$'	('date$')'newline$''newline$'

# Check how many speaker/tier names the user has filled in to the form:
numberOfSpeakers = 0
for tier from 1 to 4
	if tier_name_'tier'$ <> ""
		numberOfSpeakers = numberOfSpeakers + 1
	else
		tier = 4
	endif
```

```
endfor

# Get the correct tier index number for each tier/speaker label in the TextGrid:
for tier to numberOfSpeakers
	tiername$ = tier_name_'tier'$
	call GetTier "'tiername$'" tier_'tier'
		if tier_'tier' = 0
			exit The tier "'tiername$'" was not found in the selected
TextGrid! Please check the name.
		endif
	numberOfIntervals_'tier' = Get number of intervals... tier_'tier'
	int_'tier' = 1
	end_'tier' = start_time
endfor

# Initialize some variables
tier = tier_1
start = start_time
preceding_start = start_time - 1
preceding_end = start_time - 1
pause = 0
current_speaker = 0
start_0 = start_time
end_0 = start_time
overlapped = 0
overlapped_by = 0
line = 0
line$ = ""

# As long as new transcribed intervals are found, loop through the parallel tiers in
the TextGrid:
repeat
	diff = total_duration - start
	next_speaker = 0

	# Find out which speaker starts the next utterance:
	for speaker to numberOfSpeakers
		label$ = ""
		int_'speaker' = Get interval at time... tier_'speaker' start
		start_'speaker' = Get starting point... tier_'speaker' int_'speaker'
```

```
                end_'speaker' = Get end point... tier_'speaker' int_'speaker'
                if line = 0
                        label$ = Get label of interval... tier_'speaker' int_'speaker'

                endif
                while label$ = "" and int_'speaker' < numberOfIntervals_'speaker'
                        int_'speaker' = int_'speaker' + 1
                        label$ = Get label of interval... tier_'speaker' int_'speaker'
                        start_'speaker' = Get starting point... tier_'speaker'
int_'speaker'
                        end_'speaker' = Get end point... tier_'speaker' int_'speaker'
                endwhile
                # If this speaker begins an utterance earlier or at the same time
than the
                # previously checked speakers, make this speaker's utterance the
next line to export:
                if label$ <> "" and (start_'speaker' - start) <= diff
                        next_speaker = speaker
                        diff = start_'speaker' - start
                endif
        endfor

        if next_speaker > 0

                # Check whether a change of speaker has occurred:
                if current_speaker <> next_speaker
                        switch = 1
                        preceding_speaker = current_speaker
                        current_speaker = next_speaker
                else
                        switch = 0
                endif
                start = start_'current_speaker'
                current_end = end_'current_speaker'

                # Get the name of the new speaker, if a turn switch occurred:
                if switch = 1
                        speaker$ = tier_name_'current_speaker'$
                        speaker$ = speaker$ + ":"
                else
```

```
                speaker$ = ""
        endif

        # Get the utterance text:
        label$ = Get label of interval... tier_'current_speaker'
int_'current_speaker'

        # If the current utterance is completely overlapped by a preceding
speaker, keep this information:
        if preceding_end > current_end and preceding_start < start
                overlapped = 1
                overlapped_by = preceding_speaker
        else
                overlapped = 0
                overlapped_by = 0
        endif

        # If an utterance is completely overlapped by another, mark it in
square brackets:
        if overlapped = 1 and include_start_times_for_overlaps = 0
                label$ = "['label$']"
        # Calculate the duration of a pause during which nobody speaks:
        elsif overlapped = 0 and preceding_end < start and
insert_pause_durations = 1 and line > 0
                pause = start - preceding_end
                pause$ = "                                ('pause:2' s)" +
newline$
                fileappend "'filepath$'" 'pause$'
        # The starting point of a following overlap is marked like a pause
but as a negative number,
        # indicating the starting point of overlap relative to the end of the
preceding utterance.
        # If an utterance overlaps with several previous utterances either
partly or fully,
        # the overlap time is calculated from the overlapped utterance that
started first.
        elsif preceding_end > start and include_start_times_for_overlaps =
1 and line > 0
                pause = start - preceding_end
```

```
                pause$ = "                              ('pause:2' s)" +
newline$
                fileappend "'filepath$'" 'pause$'
                if overlapped = 1
                        label$ = "['label$']"
                endif
        endif

        # Write the utterance line to the text file and increase counter:
        if insert_start_times = 1
                line$ = "['start:2' s]      "
        endif
        line$ = line$ + "'speaker$'      'label$'"
        #printline 'line$'
        line$ = line$ + newline$
        fileappend "'filepath$'" 'line$'
        line = line + 1

    endif

    # Change the start and end time of the preceding utterance only if there
was no complete overlap:
    if overlapped = 0
        preceding_start = start
        preceding_end = current_end
    endif

    # Calculate and display progress in percent of total duration:
    percentage = start / total_duration * 100
    echo 'percentage:0' %

    # In case you want the script to stop after adding a specific number of
lines,
    # uncomment the next three lines and edit the number as required:
    #if line = 30
    #   exit Only the first 'line' lines were inserted in the text file.
    #endif

    line$ = ""
    start = start + 0.00001
```

```
until next_speaker = 0

#Genericize
echo 'line' lines of conversation were written in 'filepath$'.


#-------------
# This procedure finds the number of a tier that has a given label.

procedure GetTier name$ variable$
	numberOfTiers = Get number of tiers
	itier = 1
	repeat
		tier$ = Get tier name... itier
		itier = itier + 1
	until tier$ = name$ or itier > numberOfTiers
	if tier$ <> name$
		'variable$' = 0
	else
		'variable$' = itier - 1
	endif

endproc
```

List of Symbols, Abbreviations, and Acronyms

Ant	Antecedent
DM	Dialogue Manager
Rel	Relation
RN	Robot Navigator
SCOUT	Situated Corpus of Understanding Transactions
TU	transaction unit
WoZ	Wizard-of-Oz

1 (PDF)	DEFENSE TECHNICAL INFORMATION CTR DTIC OCA
1 (PDF)	CCDC ARL FCDD RLD CL TECH LIB
1 (PDF)	GOVT PRINTG OFC A MALHOTRA
1 (PDF)	CCDC ARL FCDD RLC IT C BONIAL