January 23, 1998

Didi User's Manual

Nigel Ward

1 Purpose and Provenance

Didi is a tool for the analysis of conversations.

It is intended to support, first, the ability for an analyst to closely examine recorded conversations using both eye and ear.

Second, it supports labeling of words and other events, as a prerequisite to the quantitative analysis of conversations, and, to a lesser extent, as the prerequisite to preparing data for training speech recognizers.

Didi was built by Nigel Ward, with contributions from Wataru Tsukahara, at the University of Tokyo, over the period from 1994–1998.

Other documentation on `didi` includes the Didi Overview, at the didi home page (http://www.sanpo.t.u-tokyo.ac.jp/∼nigel/didi/), the Didi Command Summary, `dcmds.c` in the `src` directory in the distribution, and the Didi Bug List, `bugs.txt` in the `doc` directory in the distribution.

2 Basic `didi`

When you invoke `didi` you will see a display something like that in Figure 1.

The fuzzy collections of dots represent the speech signals; with the left channel above and the right channel below.

The x-axis is time in milliseconds.

The vertical lines of dots are the *boundaries* between *regions*.

The thick horizontal line marks the *active region*; you can listen to this by pressing the '**space**' key.

If the sound is too loud or too quiet, use the `gaintool` program to set the output to "speaker" and to adjust the "play volume". (If you'd rather listen with earphones, plug them into the jack at the back of the computer and set the output to "jack".)

You can change the active region by pointing and clicking with the left mouse button. You can also move the active region with keys: forwards with the '**f**' key and backwards with the '**b**' key. You can join regions with '**m**' and split them with '**s**'.

The solid vertical line is the *active boundary*; you can move this to the left with '**l**' and to the right with '**k**'. This is useful when you want to stretch a region to correspond precisely to a single phoneme or word. You can use '**t**' to move the active boundary to the other side of the active region. The active boundary also changes when you select a new region to become the active

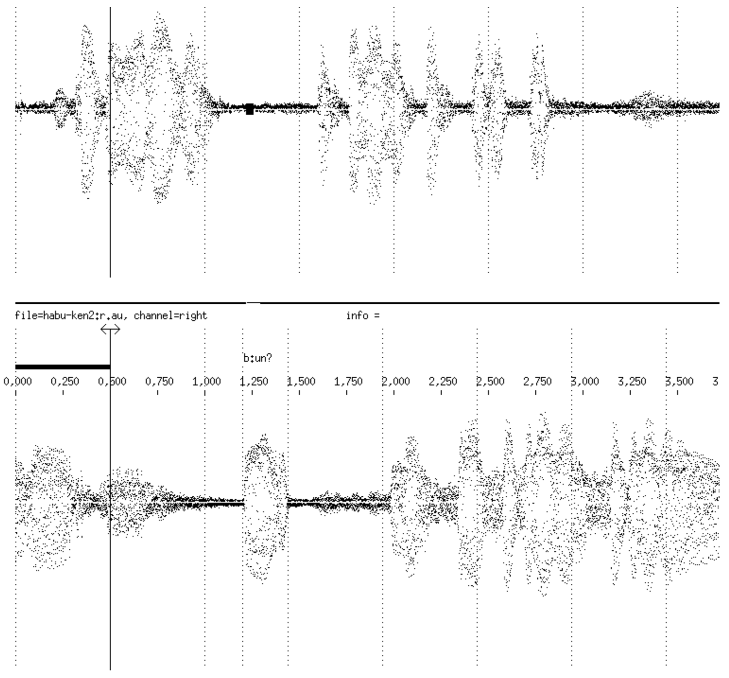

Figure 1: Initial didi display

region.

You can enter a labels for regions using the ‘**e**’ key. This draws a frame in which you can enter any label you like; hit the ‘**return**’ key when you are done. While entering or editing a label you can use ‘**c-e**’, ‘**c-a**’, ‘**c-f**’, ‘**c-b**’, ‘**c-d**’, and ‘**backspace**’; these commands have the same meanings as in `emacs`.

You can quit `didi` with ‘**q**’. When you do this it stores the labels you added (in files called xxx:l.la and xxx:r.la), and that next time you invoke `didi` , it will read in those labels. (This occasionally fails, due to a bug, so you may wnt to periodically do ‘**w**’ to write the new labels to the label files.)

Use ‘**r**’ to redraw the display.

3 Advanced `didi` Commands

The *active channel* is indicated by the position of the active region, and also by the light blue wash. To change the active channel you can select a new active region with a mouse click, or use ‘**0**’ or ‘**1**’ to specify the left or right channel.

Initially `didi` is in “stereo mode” meaning that both the left and right channels are played together. You can also choose mono mode with ‘`’`’; in this case you will only hear the active channel. You can go back to stereo mode with '`"`'.

You can play the entire audio file, starting from the active region, with the '**p**' key. You can interrupt this playback by hitting any key, and the active region will jump to the place where you stopped it.

'**d**' and '**i**' decrease and increase the timescale.

'>' and '<' scroll the display to the right and left, although scrolling is also done automatically to keep the active region visible.

'+' and '–' turn the display of prosody on and off. There is a lot of information in the prosody display, but the important thing is the display of pitch, indicated by the red curves at the bottom of each track.

'**7**' sets compact display mode (Figure 2). This is used for making print-outs of your labeling; most of the commands are inoperational in this mode. '**8**' restores normal display mode.

There are also a few minor commands; hit the '**?**' key to see the command summary.

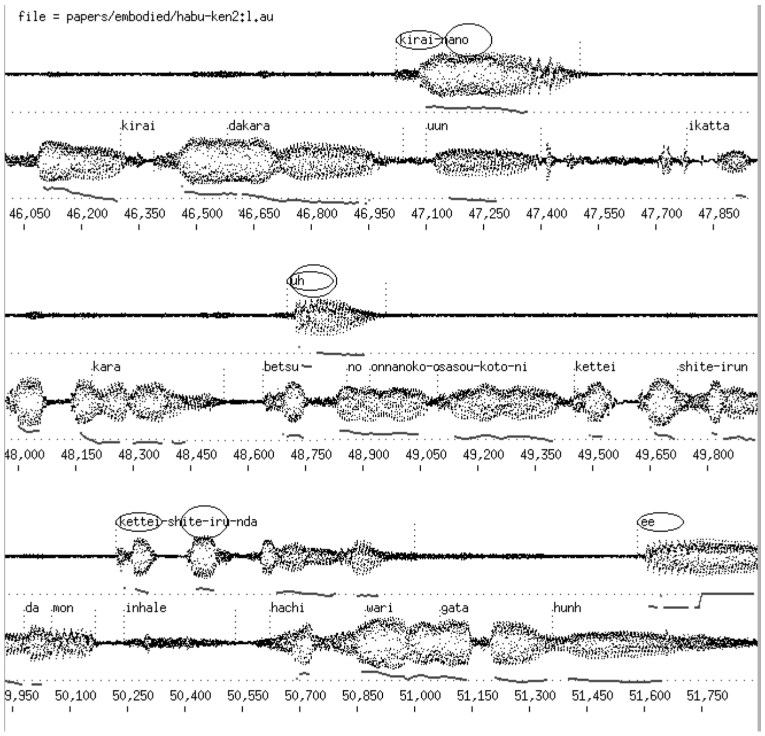

Figure 2: Didi compact display

4 Other Functionality

To visually highlight a place in the input, give it a label which begins with "a:". This will cause `didi` to underline the label in orange.

In all modes you can resize the width of the `didi` window and the display will adjust automatically;

in compact mode you can also resize the height.

As a default, `didi` initially shows the first part of the signal, but `didi -t` can be used to specify what timepoint to put at the center of the display.

Didi can be invoked in "view only mode" with the `-v` option; in this mode no labels can be edited.

More details are available with the `-h` option.

5 Satellite Programs

Didi has a bunch of little programs to help it.

`minutize` takes a long audio file (.au file), and writes it out as 60 second segments, which `didi` can process.

`soundtool` and `gaintool`, in the Sun /usr/demo/SOUND directory, are useful for recording inputs for `didi` and for setting the output volume etc. Some of this functionality is duplicated in our programs. `actl` is a general-purpose program for audio control from the command line. `to-jack` sets the audio output to go the jack. `to-speaker` sets audio output to the speaker. `grab` records a short snatch of sound from the audio input and writes it to an `.au` file.

There are also some functions for manipulating stereo files. `separate` creates two mono files from a stereo file, one for each track. `meld` creates a stereo file from two mono files. `mix` creates a mono file from two mono files.

`chopper`, given an .au file for an utterance, reads the corresponding .la file and outputs separate .au files for each word in the utterance. This is used for speech recognizer training, and for generating out-of-context samples so you can listen to them.

Finally, two specialized tools. `respond` uses prosodic information to predict the occurrence of back-channel feedback. It is included as an example of how to work with audio files without going through `didi` . `val` is used to compute the coverage and accuracy of `respond`'s predictions. It is an example of a simple program for processing `.la` (label) files.

Most of these commands have a `-h` option which explains what arguments they take.