status report for

Preparation of the JEIDA Japanese Common Speech Data Corpus

Contract No. MDA972-92-J-1016
ISIP Project No. 03-95

for the period of July 1, 1995 to September 30, 1995

submitted to:

Linguistic Data Consortium

441 Williams Hall
University of Pennsylvania
Philadelphia, PA 19104-6305

submitted by:

Joseph Picone, Ph.D., Associate Professor

Institute for Signal and Information Processing
Department of Electrical and Computer Engineering
Mississippi State University
Box 9571
216 Simrall, Hardy Rd.
Mississippi State, Mississippi 39762
Tel: 601-325-3149
Fax: 601-325-3149
email: picone@isip.msstate.edu

EXECUTIVE SUMMARY

The second phase of this project saw considerable progress in four areas: digitization, segmentation, training, and validation. Particularly, due to recent enhancements in the validation software tools, we have increased validation efficiency considerably, and hence appear to be able to complete the project significantly ahead of schedule.

Digitization of the corpus is proceeding smoothly. Due to ISIP's robust audio environment based on the Townshend Computer Tools DAT-Link+, and its vast disk resources (30 Gbytes of disk always comes in handy), we are able to transfer the data from tape to disk and perform off-line processing. This has proven handy in refining some of the supporting software. The only problem we have encountered is that some of the tapes appear to randomly produce a loss in synchronization due to tape dropouts. When digitizing these tapes, the recording system suddenly disconnects because it loses synchronization with the data stream. We are told that some DAT machines are more sensitive to dropouts than others. We are researching this problem in more detail (we think the source of the problem might be related to the JEIDA tapes), and are communicating with Professor Itahashi about it. In any event, all isolated digit phrases and four digit phrases have been digitized. This represents about 25% of the corpus.

Based on some problems we observed in the data, we have implemented a new energy-based segmenter (yet another endpointer) that closely parallels algorithms previously developed by the PI. With this code, we have reduced the error rate on segmentation to something on the order of about 0.1% of the utterances digitized. This code, as with all ISIP code, is written in C++ and will be placed into the public domain.

We have completed several passes of training for the four people involved in validation (the PI and three undergraduate students). We now have 100% agreement on a training pass of 100 utterances for the speech data, and minor variation in the level of detail supplied for non-speech sounds (mainly when to mark "significant" mouth noises). We are constantly working to improve the degree of agreement on the non-speech sounds.

Perhaps the most exciting apsect of our progress is the development of a new version of the validation tool. This tool, written almost entirely in tcl, allows users to validate data mainly from the mouse. Users select items from a listbox (menu) and can do several rudimentary file manipulations (save file, move to next utterance, and play) with a single click. Refinement of this tool has improved validation rates from 100 utterances per hour to 500 utterances per hour. This means a validator can validate an entire tape in a single session (about 2.5 hours). Though the tool is still slower than we would like due to its tcl implementation, it is now sufficiently fast to not be the bottleneck in the validation process.

Progress has also been made on the development of a standard format for the SPHERE files to be used to distribute the corpus, and a tool to generate these files from the validation data. However, we have yet to reach an industry consensus on the names and types of information to be stored in the SPHERE files.

Given the current rate of validation, we expect to achieve our goal of completing this project by the end of the year.

1. DIGITIZATION

Digitization of the corpus is proceeding smoothly. ISIP audio environment for this project is based on the Townshend Computer Tools DAT-Link+. We continue to be impressed with the robustness of these units. One and two hour tapes are routinely digitized with no real-time errors. We correctly anticipated there would be problems with the speech segmentation, so we decided to perform digitization in two steps: uploading the data to system disks, and off-line segmentation of the data.

Fortunately, because we have vast disk resources (30 Gbytes of disk always comes in handy), we are able to accomplish this off-line processing effortlessly. We currently have 25% of the corpus on-line as raw data files, and a duplicate of this data in a one utterance per file format, and we still have about 20 Gbytes of disk space free. This ability to buffer data has proven invaluable in overcoming problems with the segmentation algorithm (see Section 2). An overview of the status of digitization is given in Appendix A.

The only problem we have encountered is that we believe our copies of the tapes are flawed. We are experiencing two types of problems. Some of the tapes appear to randomly produce a loss in synchronization due to tape dropouts. When digitizing these tapes, the recording system suddenly disconnects because it loses synchronization with the data stream. We are told that some DAT machines are more sensitive to dropouts than others. We are researching this problem in more detail (we think the source of the problem might be related to the JEIDA tapes), and are also communicating with Professor Itahashi about it.

We normally digitize tapes at night, so when this happens, it costs us time the next day repairing the damage. It appears to happen on four of the first 18 tapes we have processed — about 33% of the tapes. In all cases except one, we have been able to recover the data. In one case, it appears there is data missing at the end of the tape.

The second problem we have encountered is that a few utterances in the middle of a session are garbage — indicating some type of tape decoding problem has occurred. We have observed this occasionally in DATs — the equivalent of a tape-dropout with analog tapes. The data that comes back is absolute noise (similar to what you hear when you play non-speech data into a D/A converter). We will probably request Professor Itahashi to transmit new copies of these specific utterances.

2. SEGMENTATION

We seriously tried to avoid reinventing the wheel on segmentation algorithms. We also wanted to avoid using a recognition-based approach, mainly because of the complexity of the software required to do this type of segmentation (and the fact that the undergraduate staff assigned to this project are not well-versed in recognition-based approaches). After experimenting with several algorithms, including a shareware algorithm, we found that the JEIDA corpus contains several unique characteristics that cause most unsuspecting software to fail.

First and foremost, the utterances on the tape are spaced fairly closely — many times on the order of 0.3 seconds apart. This is a less than desirable situation from a segmentation standpoint,

especially when the utterance durations range from 100 msec to 500 msec. The small utterance separation interval makes it difficult to maintain robustness to noise and non-speech sounds while providing accurate segmentation information.

Equally troublesome, however, is the fact that the average tape is a concatenation of several acoustic environments, sandwiched between an announcement indicating the speaker and session number, which was drawn from yet another ambient environment. An example of this is shown in Figure 1. Such artificial changes in the background channel, which we refer to as the background noise level, cause problems for the average segmentation algorithm that assumes a slowly-varying background noise level. Hence, sometimes, at the onset of a new session, immediately after the session introduction, the first utterance can get corrupted while the noise level adapts to the new channel condition.

Finally, one of the things that confounds the segmentation problem is the appearance of an appreciable number of non-speech sounds between shortly spaced utterances. These invariably result in the segmenter detecting these as a single utterance. Such noises, which often have energy above the background noise level cannot be ignored because they occur within a short distance of an utterance. They will typically get attached to the previous utterance, and then attached to the following utterance as well. The result is a single file containing two utterances. Fortunately, the frequency of such events appears to be approximately 2 out of 1000 utterances (a few utterances per tape), and are highly correlated with the speaker. These utterances require hand segmentation during a postprocessing quality control step.

We spent a great deal of time experimenting with different approaches to optimize performance, including some public domain software. After seeing these algorithms repeatedly fail, we finally

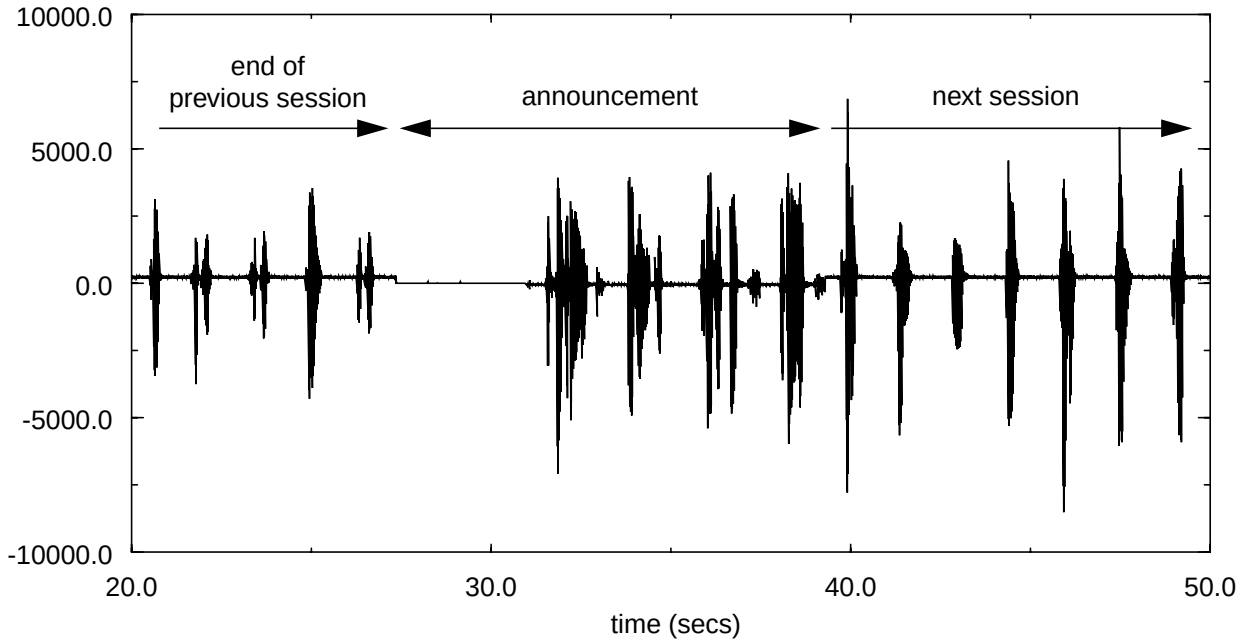

Figure 1. An example of how the JEIDA data appears in digitized form. The concatenation of the announcement, recorded under a different channel condition, often causes unpredictable behavior for standard segmentation algorithms.

decided to develop a new algorithm based on our many years of experience with this problem (we did not have any such code laying around). The goal was to produce code that could be easily optimized for the different anomalous behaviors we were encountering in the data. An overview of the parameter file used to control the algorithm is given in Figure 2. An excerpt from the class header file is given in Figure 3.

The software is written entirely in C++ and is extremely modular. This algorithm separates the segmentation problem into: prefiltering, an overlapping window-based energy computation, a four-state machine that monitors transitions between signal and noise states, a circular buffer that supports a delayed-decision strategy, and a smoothing step that discriminates between signal bursts and noise bursts. It has proven to be quite robust, and is easy to optimize for a given recording condition. We are currently observing error rates of less than 0.1% on the data. Most of the time, failures are due to real artifacts in the data, not algorithm deficiencies.

In the process of developing the segmentation code, several simple utilities to facilitate the project were added. These are all based on public domain software, and include a plot_signal program based on xmgr, which is shown in Figure 1. A very useful tool has been a program called plot_endpoints, which allows segmentation markers to be easily reviewed. An example of this program is shown in Figure 4. The net result of the combination of these utilities is that digitization and segmentation of the corpus is now a fairly automated procedure.

As an aside, we have a group of students working on a new version of the segmentation program as a course project for our introductory DSP class. The next version of this code will be even more modular, supporting run-time selection of different algorithms through a virtual class mechanism.

3. TRAINING

We have completed three passes of training for the four people involved in validation (the PI and three undergraduate students). Our procedure was to select 100 utterances at random, let each validator process these utterances, and then compare and discuss the results. Since this corpus is a fairly simple validation task (the corpus primarily consists of isolated phrases), transcription of the speech data has been straightforward (see Section 4 for more details).

We now have 100% agreement on a training pass of 100 utterances for the speech data, and minor variation in the level of detail supplied on non-speech sounds (mainly when to mark “significant” mouth noises). We are constantly working to improve the degree of agreement on the non-speech sounds. A comparison of the validators’ data for one of the most recent training passes is given in Appendix B.

4. VALIDATION

Perhaps the most exciting aspect of our progress is the development of a new version of the validation tool. This tool, written almost entirely in tcl, allows users to validate data by primarily using the mouse. A validator’s hands never need to leave the keyboard or mouse positions. Users select items from a listbox (menu) and can do several rudimentary file manipulations (save file, move to next utterance, and play) with a single click. Forcing all validators to use a predefined set symbols improves their overall consistency.

```
# file: signal_detector_00.params
#
# this file contains the parameters used to endpoint speech
# utterances recorded under near studio-quality conditions.
# it was originally developed to digitize the JEIDA CSD Corpus.
#
# this file has been “optimized” for short isolated word utterances,
# such as the isolated digits. the JEIDA data is packed quite closely,
# as little at 0.3 secs separates some utterances. so some parameters,
# such as minimum_utterance_separation, have been set very small.
#

# data format parameters
#
number_of_channels          = 2 channels
sample_size                 = 2 bytes
channel_to_be_processed     = 0 channel

# signal processing parameters
#
sample_frequency            = 16000.000 Hz
frame_duration              =    0.020 sec
window_duration             =    0.030 sec
preemphasis                 =    0.950 units

# signal level-related energy parameters
#
nominal_signal_level        = -35.00 dB
signal_adaptation_delta     = 15.00 dB
signal_adaptation_constant  =   0.50 units

# noise level-related energy parameters
#
nominal_noise_level         = -60.00 dB
noise_adaptation_delta      = 15.00 dB
noise_adaptation_constant   =   0.75 units
noise_floor                 = -70.00 dB

# utterance-related parameters
#
utterance_delta             = 6.000 dB
minimum_utterance_duration  = 0.060 sec
minimum_utterance_separation = 0.300 sec
maximum_utterance_duration  = 99.999 sec

# debug information
#
debug_level                 = 0 level

#
# end of file
```

Figure 2. An example of a parameter file used to control the energy-based segmenter. The parameters shown are those used for segmenting isolated words and syllables.

```
class Signal_detector {

public:
  // required methods
  //
  char_1* name_cc();
  logical_1 debug_cc(FILE* fp, char_1* message);
  volatile void error_handler_cc(char_1* method_name, char_1* message);

  // constructors/destructors
  //
  Signal_detector();
  ~Signal_detector();

  // i/o methods
  //
  logical_1 load_parameters_cc(char_1* param_filename);
  int_4 get_nbytes_to_read_cc();

  // initialization methods
  //
  logical_1 initialize_cc();
  logical_1 flush_cc();
  logical_1 reset_cc();

  // computational methods
  //
  logical_1 process_cc(void* signal);

  // status methods
  //
  int_4 in_progress_cc();
  logical_1 get_endpoints_cc(float_4& t1, float_4& t2);

private:
   ...
  // state machine related methods
  //
  logical_1 sm_reset_cc();
  int_4 sm_advance_cc(float_4 egy);

  // circular buffer related methods
  //
  int_4 cb_increment_cc(int_4 index, int_4 value);
  int_4 cb_diff_cc(int_4 val1, int_4 val2);
  int_4 cb_add_cc(int_4 state);
  logical_1 cb_utt_in_progress_cc();
  logical_1 cb_utt_not_in_progress_cc();
...
```

Figure 3. An excerpt from the signal_detector header file showing the public interface to the algorithm. An internal circular buffer and state machine is used to simply the algorithm implementation and to better document the many heuristics involved. This algorithm eventually will be generalized into a higher level class that supports several competing algorithms from the same programming interface.

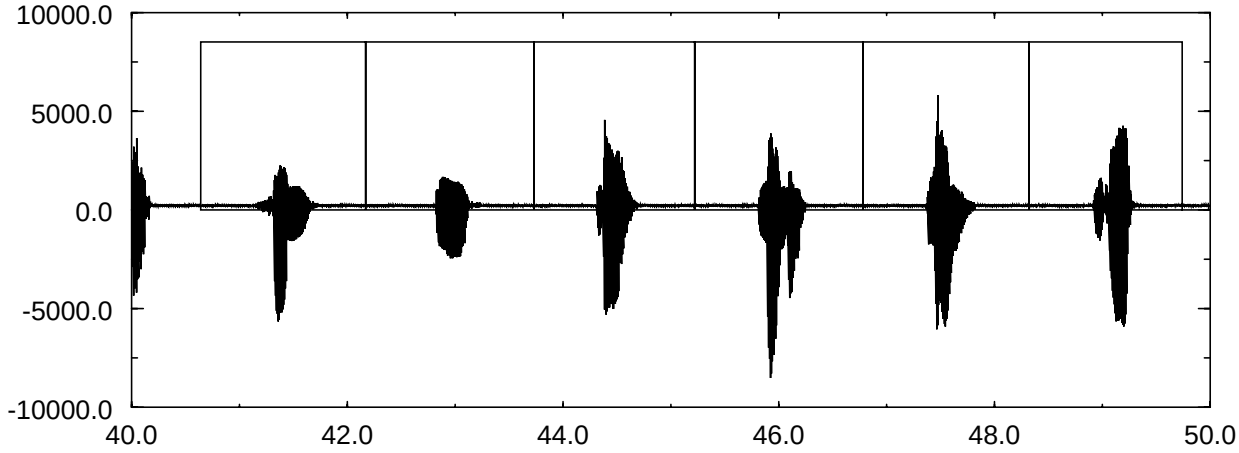

Figure 4. An example of an output from the plot_endpoints program. This program is used to review the results of the segmenter. Debugging information output from the signal_detector program can also be easily viewed using xmgr.

An overview of the validator is given in Figure 5. The functions of most of the buttons are fairly intuitive. A list of files is input to the program, and this list is manipulated as a stack by the validator. Validators can move forward, backward, or do random access into the list. Several accelerator keys have been added to reduce the amount of time validators need to spend manipulating the mouse, the keyboard, and the file list.

In its current form, the validators, in a typical cycle, simply have to hit the save-next-play (SNP) button (assuming the transcription is complete), click on the words they hear with the left mouse button, and hit SNP again. SNP can be activated from the middle mouse button, and selection of items requires only a single mouse click (not normal for tcl). Hence, users spend most of their time using the left and middle mouse buttons (or the left mouse button and the ESC key).

Refinement of this tool has improved validation rates from 100 utterances per hour to 500 utterances per hour. This means a validator can validate an entire one hour tape in a single session (about 2.5 hours). Though the tool is still slower than we would like due to its tcl implementation, it is now sufficiently fast to not be the bottleneck in the validation process. Tcl proved to be extremely inefficient — much more so than we ever imagined. We have consulted with numerous people around the world about the bottlenecks in tcl, but have yet to find a good solution to the performance bottlenecks. We rewrote code several times to minimize the amount of processing performed in tcl. The largest portion of the execution time for the tcl code occurs updating the text display — which turns out to be very expensive in tcl.

Waveform plotting was added to aid in evaluation of the segmentation, and improve the consistency of marking of the non-speech sounds., Our general rule for non-speech sounds is that if you can see them, and hear them, you should mark them. About 10% of the utterances for isolated digits end up having mouth noises, and it varies significantly with the speaker and the nature of the recording session (some speakers were clearly more careful than others).

We have followed the LDC Call Home conventions for transcriptions where appropriate. We have decided to mark nonstandard pronunciations of the words. For example “ich” and “hach” are

Figure 5. An overview of the JEIDA corpus validation tool. Users select from choices that appear in the listboxes using the mouse. Users can hear the data by using a number of play options, can view the waveform, shown at the bottom of the window, and can manipulate a validation list by rotating through a stack of files. Accelerator keys have significantly improved validation efficiency.

marked as such ("[ich]" and "[hach]" respectively). In the final SPHERE files, we will deliver multiple transcriptions, providing users with whatever level of detail they should need. From a speech recognition standpoint, it is useful to have these alternate pronunciations — since these are a real issue in connected digit recognition, and don't require significantly more effort to generate.

5. SPHERE FILE CONVERSION

We have begun designing the header to be used to capture the information available to us in the JEIDA corpus. We have distributed a document for comments/recommendations. A list of the information is shown in Table 1. A simple program has been implemented to convert the output of the validation program, a raw speech file and its associated validation file, into a SPHERE file. SPHERE software has been acquired and built under Solaris 2.4 (which required a minor modification from NIST). We don't anticipate any serious problems with this phase of the project. Given the improved throughput of the validation phase of this project, our current plan is to perform the conversions after the entire corpus has been validated. At this time, we will resolve minor problems associated with various utterances.

Object Name	Example
header_style	NIST
block_size	1024
microphone	Sanken MU-2C or Condensor
recording_site	CAN, ECL, ...
database_id	jeida common speech data
database_version	1.0
recording_environment	soundproof room,...
speaker_session_number	MS-03 (monosyllables-session 03)
speaker_id	127
speaker_sex	male
speaker_age_category	20-29
speaker_height	181 cm
speaker_original_address	Yokohama
speaker_present_address	Kanagawa
ambient_noise_level	19 dbA
speaking_mode	read
sampel_count	(automatic)
sample_min	(automatic)
sample_max	(automatic)
sample_average	37.6
sample_rate	16000
sample_n_bytes	2
sample_byte_format	linear
sample_sig_bits	16
sample_byte_coding	pcm
sample_checksum	(automatic)
prompting_text	“ichi ni san hachi”
kanji_prompting_text	EUC version of the above
orthographic_transcription	“{mouth noise} ichi ni san [hach]”

Table 1. An overview of the proposed SPHERE header for the JEIDA Common Speech Data Corpus. Several levels of transcriptions will be provided to accommodate different acoustic model training approaches.

6. NEAR-TERM PLANS

We have currently digitized approximately 25% of the corpus. Over the next phase of the contract, we expect to complete the following:

- digitization of the remainder of the data;
- validation of digits, monosyllables, and control words;
- generation and organization of the above data into the final corpus filename scheme.

Off-line, by the end of the year, we hope to complete implementation of a new version of the validation tool that removes the throughput bottlenecks we are experiencing with tcl. This exercise will serve two purposes: provide a more efficient version of the tool for distribution, and train some ISIP programmers on X windows applications programming.

We expect to have approximately 50% of the corpus validated at the time we submit the next status report. We are now entering the most time-consuming phase of validation, the four digit strings.

APPENDIX A: Validation Status

				Status		
Tape No.	ID No.	Material	Dur. (hours)	Digit.	Val.	Comments
1	1-1	Isolated Dig.	1	yes	yes	
2	1-2	Isolated Dig.	1	yes	yes	dropout
3	2-1	Isolated Dig.	1	yes	yes	
4	2-2	Isolated Dig.	1	yes	yes	
5	3-1	Isolated Dig.	1	yes	yes	
6	3-2	Isolated Dig.	1	yes	yes	
7	4-1	Isolated Dig.	1	yes	yes	
8	4-2	Isolated Dig.	1	yes	yes	
9	1	4 Digit Seq.	2	yes	no	
10	2	4 Digit Seq.	2	yes	no	dropout at end of tape
11	3	4 Digit Seq.	2	yes	no	
12	4	4 Digit Seq.	2	yes	no	
13	5	4 Digit Seq.	2	yes	no	
14	6	4 Digit Seq.	2	yes	no	dropout at 1 hr. 50 min.
15	7	4 Digit Seq.	2	yes	no	dropout at 23 min.
16	8	4 Digit Seq.	2	yes	no	
17	9	4 Digit Seq.	2	yes	no	dropout near end of tape
18	10	4 Digit Seq.	1	yes	no	
19	1	Monosyll.	2	no	no	
20	2	Monosyll.	2	no	no	
21	3	Monosyll.	2	no	no	
22	4	Monosyll.	2	no	no	
23	5	Monosyll.	2	no	no	
24	6	Monosyll.	2	no	no	
25	7	Monosyll.	2	no	no	
26	8	Monosyll.	1	no	no	
27	9	Monosyll.	2	no	no	
28	10	Monosyll.	2	no	no	
29	11	Monosyll.	2	no	no	
30	12	Monosyll.	2	no	no	
31	13	Monosyll.	2	no	no	
32	14	Monosyll.	2	no	no	
33	15	Monosyll.	2	no	no	
34	16	Monosyll.	1	no	no	

				Status		
Tape No.	ID No.	Material	Dur. (hours)	Digit.	Val.	Comments
35	1-1	Ctrl. Words A	1	no	no	
36	1-2	Ctrl. Words A	1	no	no	
37	2-1	Ctrl. Words A	1	no	no	
38	2-2	Ctrl. Words A	1	no	no	
39	3-1	Ctrl. Words A	1	no	no	
40	3-2	Ctrl. Words A	1	no	no	
41	4-1	Ctrl. Words A	1	no	no	
42	4-2	Ctrl. Words A	1	no	no	
43	1-1	Ctrl. Words B	1	no	no	
44	1-2	Ctrl. Words B	1	no	no	
45	2-1	Ctrl. Words B	1	no	no	
46	2-2	Ctrl. Words B	1	no	no	
47	3	Ctrl. Words B	1	no	no	
48	4-1	Ctrl. Words B	1	no	no	
49	4-2	Ctrl. Words B	1	no	no	
50	5-1	Ctrl. Words B	1	no	no	
51	5-2	Ctrl. Words B	1	no	no	
52	6	Ctrl. Words B	1	no	no	
53	1	Ctrl. Words C	2	no	no	
54	2	Ctrl. Words C	2	no	no	
55	3	Ctrl. Words C	2	no	no	
56	4	Ctrl. Words C	2	no	no	
57	5	Ctrl. Words C	2	no	no	
58	6	Ctrl. Words C	2	no	no	

				Status		
Tape No.	ID No.	Material	Dur. (hours)	Digit.	Val.	Comments
59	1	City Names	2	no	no	
60	2	City Names	2	no	no	
61	3	City Names	2	no	no	
62	4	City Names	2	no	no	
63	5	City Names	2	no	no	
64	6	City Names	2	no	no	
65	7	City Names	2	no	no	
66	8	City Names	2	no	no	
67	9	City Names	1	no	no	
68	10	City Names	2	no	no	
69	11	City Names	2	no	no	
70	12	City Names	2	no	no	
71	13	City Names	2	no	no	
72	14	City Names	2	no	no	
73	15	City Names	2	no	no	
74	16	City Names	2	no	no	
75	17	City Names	2	no	no	
76	18	City Names	1	no	no	

APPENDIX B: Results of a recent validation training pass.

		Validator			
File	Answer	No. 1	No. 2	No. 3	No. 4
300	[hach]	[hach]	[hach]	[hach]	[hach]
301	[shich]	[shich]	[shich]	[shich]	[shich]
302	kyu	kyu	kyu	kyu	kyu
303	[ich]	[ich]	[ich]	[ich]	[ich]
304	<garbage>	<garbage>	<garbage>	<garbage>	<garbage>
305	<garbage>	<garbage>	<garbage>	<garbage>	<garbage>
306	<garbage>	<garbage>	<garbage>	<garbage>	<garbage>
307	<garbage>	<garbage>	<garbage>	<garbage>	<garbage>
308	zero	zero	zero	zero	zero
309	san	san	san	san	san
310	ni	ni	ni	ni	ni
311	rei	rei	rei	rei	rei
312	nana	nana	nana	nana	nana
313	yon	yon	yon	yon	yon
314	go	go	go	go {breath noise}	go
315	maru	maru	maru	maru {breath noise}	maru
316	shi	shi	shi	shi {breath noise}	shi
317	roku	roku	roku	{mouth noise} roku	roku
318	ku	ku	ku	ku	ku
319	hachi	hachi	hachi	hachi	hachi {breath noise}
320	shichi	shichi	shichi	shichi	shichi
321	kyu	kyu	kyu	ku	kyu
322	ichi	ichi	ichi	ichi {breath noise}	ichi {breath noise}
323	<garbage>	<garbage>	<garbage>	<garbage>	<garbage>
324	<garbage>	<garbage>	<garbage>	<garbage>	<garbage>
325	<garbage>	<garbage>	<garbage>	<garbage>	<garbage>
326	<garbage>	<garbage>	<garbage>	<garbage>	<garbage>
327	zero	zero	zero	zero	zero
328	san	san	san	san	san
329	ni	ni	ni	ni {breath noise}	ni
330	rei	rei	rei	rei	rei
331	nana	nana	nana	nana	nana

		Validator			
File	Answer	No. 1	No. 2	No. 3	No. 4
332	yon	yon	yon	yon	yon {mouth noise}
333	go	go	go	go {breath noise}	go
334	maru	maru	maru	maru {breath noise}	maru {breath noise}
335	shi	shi	shi	shi	shi {mouth noise}
336	roku	roku	roku	roku {breath noise}	roku {mouth noise}
337	ku	ku	ku	ku	ku
338	hachi	hachi	hachi	hachi	hachi
339	shichi	shichi	shichi	shichi	shichi
340	kyu	kyu	kyu	kyu	kyu {breath noise}
341	ichi	ichi	ichi	ichi {breath noise}	ichi {breath noise}
342	<garbage>	<garbage>	<garbage>	<garbage>	<garbage>
343	<garbage>	<garbage>	<garbage>	<garbage>	<garbage>
344	<garbage>	<garbage>	<garbage>	<garbage>	<garbage>
345	<garbage>	<garbage>	<garbage>	<garbage>	<garbage>
346	zero	zero	zero	zero	zero {breath noise}
347	san	san	san	san	san {breath noise}
348	ni	ni	ni	ni {breath noise}	ni {breath noise}
349	rei	rei	rei	rei	rei {breath noise}
350	nana	nana	nana	nana {breath noise}	nana {breath noise}
351	yon	yon	yon	yon	yon
352	go	go	go	go	go {breath noise}
353	maru	maru	maru	maru {breath noise}	maru {breath noise}
354	shi	shi	shi	shi {breath noise}	shi
355	roku	roku	roku	roku	roku
356	ku	ku	ku	ku {breath noise}	ku {breath noise}
357	hachi	hachi	hachi	hachi	hachi
358	shichi	shichi	shichi	shichi {breath noise}	shichi {breath noise}
359	kyu	kyu	kyu	kyu	kyu
360	ichi	ichi	ichi	ichi {breath noise}	ichi {breath noise}
361	<garbage>	<garbage>	<garbage>	<garbage>	<garbage>
362	<garbage>	<garbage>	<garbage>	<garbage>	<garbage>
363	<garbage>	<garbage>	<garbage>	<garbage>	<garbage>
364	<garbage>	<garbage>	<garbage>	<garbage>	<garbage>
365	{mouth noise} zero	{mouth noise} zero	{mouth noise} zero	{breath noise} zero	zero

		Validator			
File	Answer	No. 1	No. 2	No. 3	No. 4
366	san	san	san <needs_review>	san	san
367	ni	ni	ni	ni	ni
368	rei	rei	rei	{mouth noise} rei	rei
369	nana	nana	nana	nana {breath noise}	nana {breath noise}
370	yon	yon	yon	yon	yon
371	go	go	go	go {breath noise}	go {breath noise}
372	maru	maru	maru	maru {breath noise}	maru {breath noise}
373	shi	shi	shi	shi {breath noise}	shi {breath noise}
374	roku	roku	roku	roku {breath noise}	roku
375	ku	ku	ku	ku {breath noise}	ku
376	hachi	hachi	hachi	hachi	hachi
377	shichi	shichi	shichi	shichi {breath noise}	shichi {breath noise}
378	kyu	kyu	kyu	kyu	kyu {breath noise}
379	ichi	ichi	ichi	ichi {breath noise}	ichi
380	<garbage>	<garbage>	<garbage>	<garbage>	<garbage>
381	<garbage>	<garbage>	<garbage>	<garbage>	<garbage>
382	<garbage>	<garbage>	<garbage>	<garbage>	<garbage>
383	<garbage>	<garbage>	<garbage>	<garbage>	<garbage>
384	zero	zero	zero	zero	zero
385	san	san	san	san	san
386	ni	ni	ni	ni	ni
387	rei	rei	rei	rei {mouth noise}	rei {mouth noise}
388	nana	nana	nana	nana	nana
389	yon	yon	yon	yon	yon
390	go	go	go	go	go
391	maru	maru	maru	maru	maru
392	shi	shi	shi	shi	shi {breath noise}
393	roku	roku	roku	roku	roku
394	ku	ku	ku	ku	ku {breath noise}
395	hachi	hachi	hachi	hachi	hachi
396	shichi	shichi	shichi	shichi	shichi
397	kyu	kyu	kyu	kyu	kyu
398	ichi	ichi	ichi	ichi	ichi
399	<garbage>	<garbage>	<garbage>	<garbage>	<garbage>