Speech Segmentation and Labeling on the NeXT Machine

Chuck Wooters and Nelson Morgan[1]

TR-90-002

January 24, 1990

Abstract

We are attempting to incorporate connectionist models into speech recognition algorithms. Since these models require a large amount of training data, it was necessary to build an automated speech labeling/segmentation application. There were two significant system requirements for this program:

- Digital-to-analog capabilities.
- Support for speedy development of applications requiring a user-interface.

The NeXT machine fulfills both of these requirements. It has built in AD/DA capabilities. Its object-oriented programming environment and application-building modules permit quick program development.

We report here on a program we have developed to integrate automatic labeling and segmentation of continuous speech with a manual system for observing and correcting these signal annotations. The overall system has functioned well enough to permit easy user marking of 600 sentences in a reasonable amount of time.

[1] International Computer Science Institute

Speech Segmentation and Labeling on the NeXT Machine

Chuck Wooters and Nelson Morgan

International Computer Science Institute
1947 Center Street, Suite 600
Berkeley, CA 94704-1105, USA

1. Background

Our current research direction combines *Multilayer Perceptrons (MLP)* with traditional *Hidden Markov Models (HMM)* in order to recognize continuous speech [Bourlard et al, 1989; Bourlard & Morgan, 1989, 1990]. We are using a large standard database of speech. The speaker dependent portion of the Resource Management database [Price et al, 1988] consists of twelve different speakers each producing 600 sentences. The acoustic waveform for each sentence has been processed by our collaborators at SRI [Murveit et al, 1988] to extract features for each consecutive 10 millisecond frame of speech. In this case, the features are discrete indices into tables (codebooks) of mel-cepstrum, differential mel-cepstrum, power, and differential power (see the Murveit reference for a detailed description).

The MLP is trained with a simple variant of the back propagation learning algorithm [Rumelhart et al, 1986; Morgan&Bourlard 1989]. Since this is a supervised learning algorithm, training requires knowledge of the desired output for every input. Each input to the MLP is a sequence of features from each 10 millisecond frame in a sentence. The desired output is the phoneme (i.e. [a] or [t], etc.) which corresponds to the input frames. In general, this information is not provided with a speech database, since the dominant techniques for speech recognition (e.g., HMMs) only require a phonetic transcription (i.e., an ordered list of phonemes) for the sentence rather than phonetic labels for each individual frame. This was the motivation behind the construction of a speech segmentation and labeling system. In particular, we needed a program to permit the convenient hand-marking of the phonetic labels and segment boundaries for continuous speech waveforms. However, it has recently become clear that the MLP may be trained starting with only a phonetic transcription for each sentence [Morgan and Bourlard, 1990]. In this approach, frame labels are generated by an automatic procedure, and hand-marking is not needed. However, phonetically accurate frame labels are still required to test the fidelity of automatic frame labeling methods. Furthermore, convenient display and playback facilities are required to examine the markings chosen by the automatic segmenter.

To obtain the desired training information (the phonetic label) for each frame of input speech, we need to know two things about a sentence:

(1) Segmentation: where each phoneme in the sentence starts and stops.

(2) Labeling: what phoneme label corresponds to each segment in the sentence (i.e., the phonetic transcription).

The only available information was the digitized data for each sentence, the set of four features for each frame in the sentence, and the English text for each sentence. This report describes the software system we have designed and implemented on the NeXT machine to generate the necessary phonetic information given this limited information.

2. Why the NeXT machine?

In order to accurately label a speech waveform, one must be able to hear it as well as see it. The NeXT machine has built in digital-to-analog capabilities. It also has an object-oriented approach to programming and a simple way of creating a user interface. The object libraries that come with the next machine include an object called a "soundview". A soundview object lets you display a waveform in a scrollable window on the NeXT screen; it also includes many methods for manipulating waveforms. One of the most useful of these methods allows you to listen to the waveform through the digital-to-analog converter. Other methods allow for scrolling the waveform in a window and scaling (zooming in or out.)

3. Development

To construct the speech labeling system, we began with a sample program that was provided with the NeXT machine. This program, called "SoundEditor", allows for the recording, displaying, editing and playback of sounds from the microphone. It also contained examples showing how to use the digital-to-analog converter and various other input/output operations.

3.1. Goals

For accurate segmentation, we required a system in which the labels could be graphically aligned to the waveform. This was important so that during scrolling the labels and waveform would move together. It was also important for saving and reading previously saved label files. We also needed the ability to

- change the size of labels
- change the name of labels
- add and delete labels
- read labels from a file
- save labels to a file
- listen to a single label

Additionally, we needed to provide automatic procedures to generate phonetic transcriptions from text, and to iteratively converge on a waveform segmentation given the transcriptions and a set of discrete input features. These programs were written in C, and were run separately from the NeXT code described here. They are briefly described in section 4.

3.2. Programming on the NeXT Machine

3.2.1. Objective C

Objective C was used to program the LabelEditor. It is an object-oriented language which is supplied by NeXT. The use of an object-oriented design enabled us to work more quickly. Most of the objects we needed were supplied as part of a NeXT library. Additionally, Objective C supports standard C, so little time was spent learning new syntax.

The main object in the LabelEditor system is called a "ScrollingSound" object. This object contains information about the name of the file which contains the waveform as well as methods for displaying a waveform in a scrollable window, zooming in and out on the waveform, playing all or just part of the waveform etc. Each waveform which is to be labeled is represented by a single ScrollingSound object.

The other main object is called a "Label" object. This object contains the information about its starting and ending location in the waveform, its name, and pointers to the previous and the next label in the waveform. Additionally, it has methods for querying and setting all of these variables.

Each ScrollingSound object will contain a pointer to a "RootLabel" object (which is a subclass of the Label object.) The RootLabel contains information about the name of the file which is to be used for saving the label information. It also contains routines for drawing itself. The RootLabel and Label objects together form a doubly linked list which is used to hold the label information for the whole waveform. So, each waveform contains one RootLabel and many Labels.

3.2.2. Interface Builder

NeXT provides a utility called "Interface Builder" which allows the separation of the design of the user interface from the design of the internal workings of a program. Using the mouse, buttons and sliders can be dragged into windows to create a graphical user interface. The operation is very similar to using a paint program to create drawings. Once the user interface has been created, the buttons and sliders can be linked to the created objects. Once they are linked, a push of the button will send a message to the designated object.

The Interface Builder permits a nice design style in which the user interface is decoupled from the internal functions of the code. The interface may be designed first, providing a top-down modularity that can be an extremely natural way of representing a task. Furthermore, the details of the interface are hidden from the programmer. This permits robust and versatile code to be written quickly.

3.3. NeXT Machine limitations

There were several problems with using the NeXT machine that we had to overcome.

3.3.1. Digital-to-analog converter

Three sampling frequencies are supported on the NeXT machine: 8 kHz, 22 kHz, and 44 kHz. The Resource Management database was sampled at 16 kHz. The play method had to be modified to include interpolation, filtering, and decimation routines so that the signal was upsampled to 22 kHz (actually 21.3 kHz, using a 4/3 rate change) before being sent to the digital-to-analog converter. It would have been preferable for NeXT to have provided more granularity in the usable sampling frequencies, although the sampling rate conversion is not extremely time-consuming. For applications in which more accurate approximation was required, or for which the time delay was an issue, this might be a more significant limitation.

3.3.2. Poor early documentation

At the time when we began development, we were using version 0.9 of the NeXT operating system. Much of the documentation for version 0.9 was not available. This was quite frustrating. Hopefully this has been corrected in version 1.0.

3.3.3. System speed

System interaction (opening new windows, etc.) appears to be maddeningly slow on the NeXT. Again, there is hope that this may be due to infant problems in the system software.

4. Using the Labeling System

The segmentation program is called "LabelEditor". It must be started from a terminal window. When the program starts up, the user is presented with a main menu and a control panel. The control panel (Fig. 1) contains two parts. The upper part contains buttons which affect the whole waveform. The lower part contains buttons which pertain only to labels. The buttons on the control panel control features such as listening to all or part of the waveform, labeling a section of the waveform, changing the name of a label, changing the size of a label, etc.

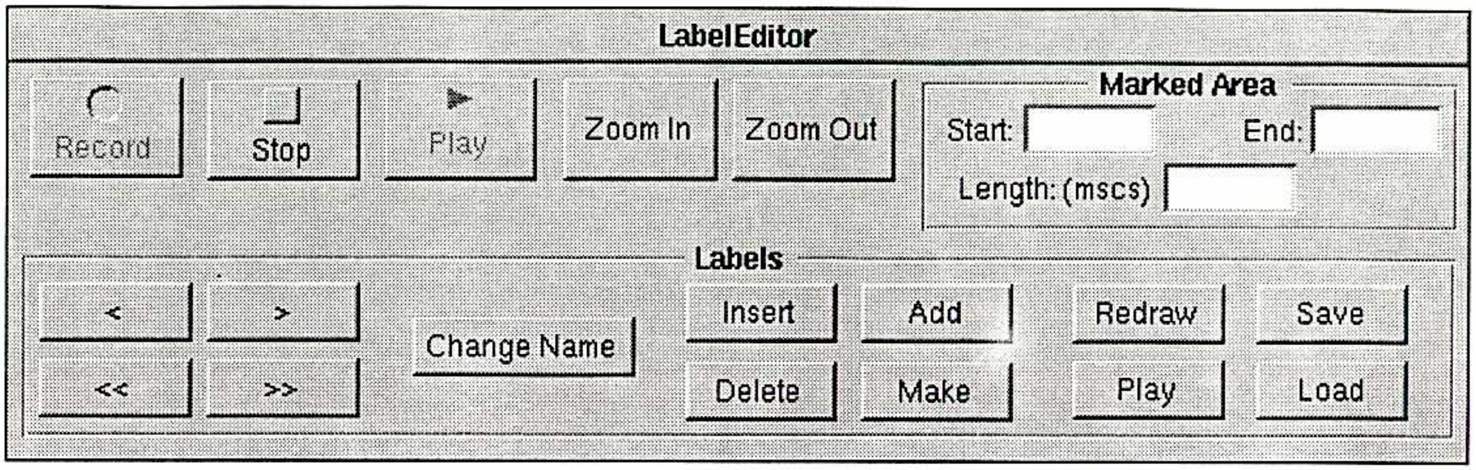


Fig. 1

From the main menu the user can open a waveform window (Fig. 2). Multiple waveforms can be displayed at the same time. Once the waveform is on the screen, the user may begin to attach labels. The labeling may be done completely from scratch or by opening a previously created label file. The labels are overlayed on the waveform display and may be edited using the mouse and keyboard.

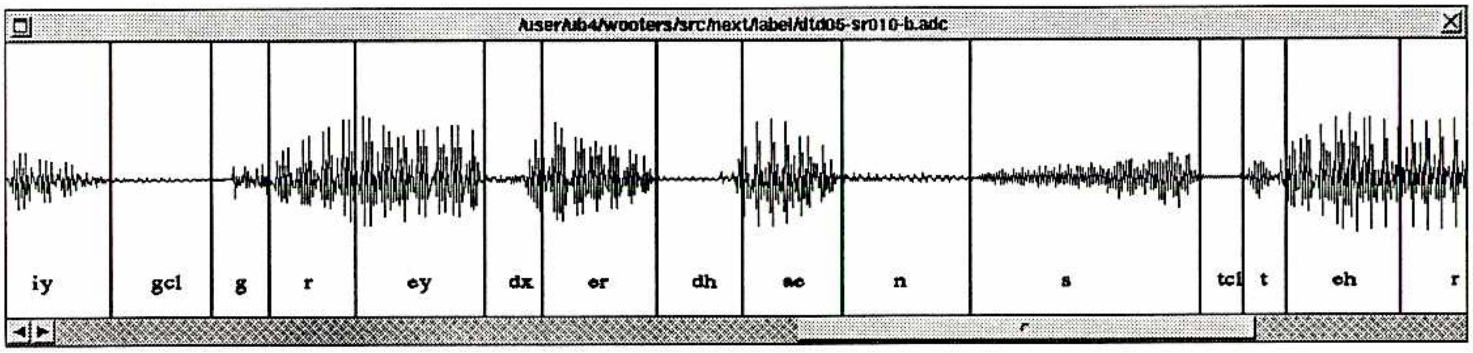
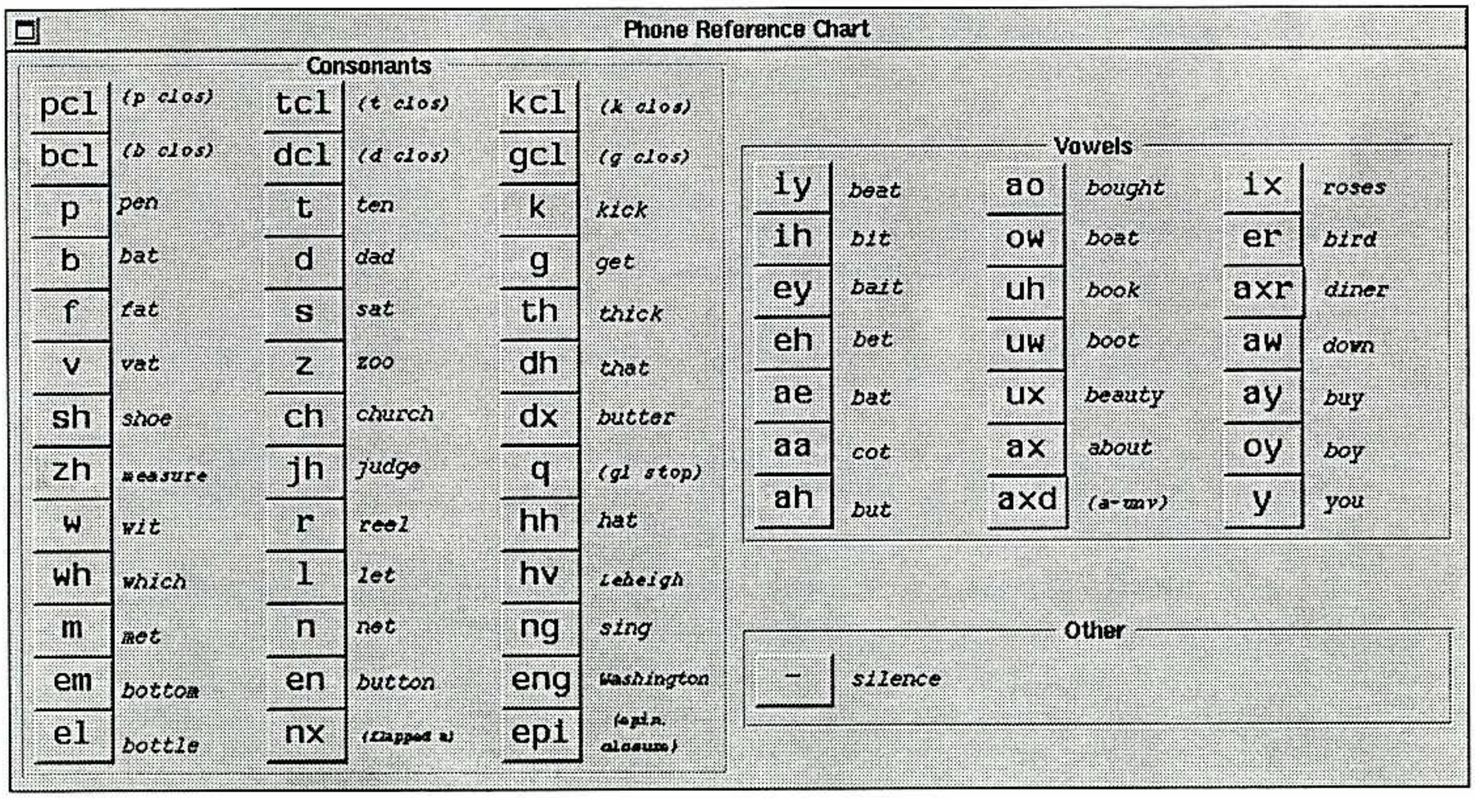


Fig. 2

As a practical matter, it is much faster to correct a moderately faulty segmentation than to mark one from scratch. What we now do is generate a transcription from a simple set of text-to-phoneme rules[1], and use this transcription and an original assumption about average phoneme durations to initialize an iterative automatic segmentation using the Viterbi algorithm, which is guaranteed to converge [Brown et al, 1983]. In the automatic segmentation, some original segmentation (e.g., one based on default lengths for each phoneme) is used to estimate conditional densities for the discrete features (given each state of a 3-state Hidden Markov Model for each phoneme). The negative logs of these estimates are used as costs for a dynamic programming solution which determines the best segmentation of the data [Ney, 1984].[2] This segmentation is then used to estimate probabilities, and so on. In practice, the process generally converges in ten or less iterations. The resulting automatic segmentation can be generated in only a few minutes on a SparcStation, and is sufficiently close to expert hand-marked segmentation to either be used with no further change, or to be refined in a relatively fast interactive step.

The last major window the user sees is a phone reference chart. (Fig. 3) This is a large window containing one button for each of the 61 possible labels. Along with the phone label is an example word containing that sound. The new user who is unfamiliar with the alphabet may click on any of the labels and hear (through the digital-to-analog converter) an example of that phone. For the more experienced user, the chart serves as an online reminder of the set of phones.



Phone Reference Chart

Consonants

pcl	*(p clos)*	tcl	*(t clos)*	kcl	*(k clos)*
bcl	*(b clos)*	dcl	*(d clos)*	gcl	*(g clos)*
p	*pen*	t	*ten*	k	*kick*
b	*bat*	d	*dad*	g	*get*
f	*fat*	s	*sat*	th	*thick*
v	*vat*	z	*zoo*	dh	*that*
sh	*shoe*	ch	*church*	dx	*butter*
zh	*measure*	jh	*judge*	q	*(gl stop)*
w	*wit*	r	*reel*	hh	*hat*
wh	*which*	l	*let*	hv	*Lehigh*
m	*met*	n	*net*	ng	*sing*
em	*bottom*	en	*button*	eng	*Washington*
el	*bottle*	nx	*(flapped n)*	epi	*(epin. closure)*

Vowels

iy	*beat*	ao	*bought*	ix	*roses*
ih	*bit*	ow	*boat*	er	*bird*
ey	*bait*	uh	*book*	axr	*diner*
eh	*bet*	uw	*boot*	aw	*down*
ae	*bat*	ux	*beauty*	ay	*buy*
aa	*cot*	ax	*about*	oy	*boy*
ah	*but*	axd	*(a-unv)*	y	*you*

Other

-	*silence*

Fig. 3

[1]The program used for this conversion was a public domain program which we modified to work with our set of phonemes. Due to the simplicity of the program, many of the words had to be hand transcribed. Additionally, the program was not sophisticated enough to deal with between-word co-articulation effects. With a better text-to-phoneme program, the number of errors produced by the automatic segmentation should be reduced significantly.

[2]The fundamental step in this algorithm, as in all dynamic programming schemes, is to compute a global cost as the lowest sum of a local cost (e.g, negative likelihood that observed features came from a given phonemic model) plus the global cost of a previous state, plus any associated transition costs, over all legal antecendent states. At each stage in the algorithm, including the last, one knows the best current total cost and can easily backtrack to find the state transition path which led to that score.

For some of our experiments, we required frame labelings which were as accurate as was practically possible. Therefore, we used a combination of automatic and manual marking procedures to generate phonetic segmentations. We have now marked 600 sentences (roughly 20,000 labels over 300,000 frames) using this system. Though still boring, it is a practical working procedure.

5. Conclusion

While there were some initial problems which had to be solved in order to use the NeXT machine, it proved to be very useful for this task. We were able to build a complete robust working system in a few weeks. The built-in digital-to-analog converter saved the cost of purchasing extra hardware (and integrating it into our system). The object-oriented programming environment, including the interface builder, made development efficient and allowed for easy modification of the code. The resulting system functioned well enough to permit easy user marking of 600 test sentences in a reasonable amount of time. However, most other speech programming efforts are being continued on another workstation (the Sun SparcStation) because of the current speed difficulties with the NeXT, and also because sophisticated analysis and display software are currently more available for the Sun than for the NeXT.

REFERENCES

Bourlard, H., Morgan, N., and Wellekens, C., ''Statistical Inference in Multilayer Perceptrons and Hidden Markov Models with Applications in Continuous Speech Recognition'', in *Neuro Computing, Algorithms, Architectures and Applications,* NATO ASI Series, 1989

Bourlard, H., and Morgan, N., ''Merging Multilayer Perceptrons & Hidden Markov Models: Some Experiments in Continuous Speech Recognition'' ICSI Technical Report TR-089-033 (1989)

Bourlard, H., and Morgan, N., ''Merging Multilayer Perceptrons & Hidden Markov Models: Some Experiments in Continuous Speech Recognition'' in *Artificial Neural Networks: Advances and Applications,* North Holland press, 1990, E. Gelenbe editor

Brown, P., Lee, C., and Sporher, J., ''Bayesian Adaptation in speech recognition'', in *Proc. IEEE Intl. Conf. on Acoustics, Speech, & Signal Processing,* pp. 761-764, Boston, MA, 1983.

Morgan, N., and Bourlard, H., ''Generalization and Parameter Estimation in Feedforward Nets: Some Experiments'' ICSI Technical Report TR-089-017 (1989) also to be published in *Advances in Neural Information Processing Systems II,* Morgan Kaufmann, 1990

Morgan, N., and Bourlard, H., ''Continuous Speech Recognition Using Multilayer Perceptrons with Hidden Markov Models'', In Press, *Proc. IEEE Intl. Conf. on Acoustics, Speech, & Signal Processing,* Albuquerque, New Mexico, 1990.

Murveit, H., and Weintraub, M., ''1000-Word Speaker-Independent Continuous-Speech Recognition Using Hidden Markov Models'', *Proc. IEEE Intl. Conf. on Acoustics, Speech, & Signal Processing,* Vol. 1, pp. 115-118, New York, 1988.

Ney, H., ''The use of a one-stage dynamic programming algorithm for connected word recognition'', *IEEE Trans. ASSP,* vol. 32, pp.263-271, 1984.

Price, P., Fisher, W., Bernstein, J., and Pallet, D., "The DARPA 1000-Word Resource Management Database for Continuous Speech Recognition", *Proc. IEEE Intl. Conf. on Acoustics, Speech, & Signal Processing*, Vol. 1, pp. 651-654, New York, 1988.

Rumelhart, D.E., Hinton, G.E. & Williams, R.J. (1986). "Learning Internal Representations by Error Propagation", in *Parallel Distributed Processing. Exploration of the Microstructure of* Cognition. Vol. 1: Foundations, Ed. D.E.Rumelhart & J.L.McClelland, MIT Press, 1986