Computer Science 401 — 13 March 2016
St. George Campus — University of Toronto

Homework Assignment #3
Due: Friday, 8 April 2016 at 19h00, electronically

Speech

TA: Arvie Frydenlund (`t3fryden@cdf.utoronto.ca`), Hengwei Guo (`t4guohen@cdf.utoronto.ca`), and Mengye Ren (`t5renmen@cdf.utoronto.ca`, marking).

1 Introduction

This assignment introduces you to Gaussian mixture modelling, continuous HMMs, and two basic tasks in speech technology: *speaker identification*, in which we try to determine *who* is talking, and *speech recognition*, in which we try to determine *what* was said.

The assignment is divided into three sections. In the first, you will experiment with speaker identification by training mixtures of Gaussians to the acoustic characteristics of individual speakers, and then identify unknown speakers based on these models. In the second section, you will experiment with speech recognition by training hidden Markov models, and analyzing your system's performance with word-error rates. In the third, you will use Watson's speech-to-text and text-to-speech services.

You will need access to both of the following data sets for each section:

1. **Training**: This contains 30 directories (e.g., `FCJF0/, FDML0/,...`), each representing a unique speaker. Each speaker speaks 9 utterances.
2. **Testing**: This contains 1 directory of 30 utterances, each spoken by a unique but unknown speaker from among the speakers in the training set.

Every utterance is represented by a set of 5 files (i.e., `*.wav, *.mfcc, *.phn, *.txt, *.wrd`); these formats are described in Appendix A.

You will also need the CMU dictionary, which contains the phonetic expansions of some common words.

All of the data can be found on CDF in */u/cs401/speechdata*.

---

2 Speaker Identification

Speaker identification is the task of correctly identifying speaker s_c from among S possible speakers $s_{i=1..S}$ given an input speech sequence X, consisting of a succession of d-dimensional real vectors. In the interests of efficiency, $d = 14$ in this assignment. Each vector represents a small 16 ms unit of speech called a *frame*. Speakers are identified by training data that are ascribed to them. This is a discrete classification task (choosing among several speakers) that uses continuous-valued data (the vectors of real numbers) as input.

Gaussian Mixture Models

Gaussian mixture models are often used to generalize models from sparse data. They can tightly constrain large-dimensional data by using a small number of components but can, with many more components, model arbitrary density distributions. Sometimes, they are simply used because the domain being modelled appears to have multiple modes.

Given M components, GMMs are modelled by a collection of parameters, $\theta = \{\omega_{m=1..M}, \mu_{m=1..M}, \Sigma_{m=1..M}\}$, where ω_m is the probability that an observation is generated by the m^{th} component. These are subject to the constraint that $\sum_m \omega_m = 1$ and $0 \leq \omega_m \leq 1$. Each component is a multivariate Gaussian distribution, which is characterized by that component's mean, μ_m, and covariance matrix, Σ_m. For reasons of computational efficiency, we will reintroduce some independence assumptions by assuming that every component's covariance matrix is diagonal, i.e.:

$$\Sigma_m = \begin{pmatrix} \Sigma_m[1] & 0 & \cdots & 0 \\ 0 & \Sigma_m[2] & \cdots & 0 \\ \vdots & & & \\ 0 & 0 & \cdots & \Sigma_m[d] \end{pmatrix}$$

for some vector $\vec{\sigma}^2_m$. Therefore, only d parameters are necessary to characterize a component's (co)variance.

Given these parameters, the probability of generating a particular vector, $\vec{v}$, is:

$$p\left(\vec{v}|\theta\right) = \sum_{m=1}^{M} \omega_m b_m(\vec{v}) \tag{1}$$

where b_m is defined in Appendix B.

2.1 Training Gaussian Mixture Models [10 marks]

Your first task will be to train one M-component Gaussian mixture model (GMM) for each of the speakers in the **Training** data set. Specifically, for each speaker s, train the parameters $\theta_s = \{\omega_{m=1..M}, \mu_{m=1..M}, \Sigma_{m=1..M}\}$ according to the method described in Appendix B. In all cases, assume that covariance matrices Σ_m are diagonal. Start with $M = 8$. You'll be asked to experiment with that in Section 2.3.

Your Matlab code should include a function `gmmTrain` which implements Algorithm 1 to train a GMM for every speaker in succession. Note that the parameter ϵ is adjustable. See Appendix B for additional details.

```
Input: MFCC data $X$
begin
    Initialize $\theta$
    $i := 0$
    $prev\_L := -\infty$ ; $improvement = \infty$
    while $i =< MAX\_ITER$ and $improvement >= \epsilon$ do
        $L :=$ ComputeLikelihood $(X, \theta)$
        $\theta :=$ UpdateParameters $(\theta, X, L)$ ;            // These two functions can be combined
        $improvement := L - prev\_L$
        $prev\_L := L$
        $i := i + 1$
    end
end
```

Algorithm 1: GMM training algorithm.

2.2 Classification with Gaussian Mixture Models [10 marks]

Given a set of trained parameters θ_s for speaker s, assume that the log likelihood of a test sequence $\tilde{X}$ being uttered by that speaker is:

$$\log P\left(\tilde{X};\theta_s\right) = \sum_{t=1}^{T} \log p\left(\vec{x}_t;\theta_s\right) \tag{2}$$

Your task is to classify each of the test sequences in the **Testing** data set according to the most likely speaker, $\hat{s}$:

$$\hat{s} = \underset{s=1,\ldots,S}{\operatorname{argmax}} \ \log P\left(\tilde{X};\theta_s\right) \tag{3}$$

Write a script, `gmmClassify.m`, that calculates and reports the likelihoods of the five most likely speakers for each test utterance. Put these in files called `unkn_`N`.lik` for each test utterance N.

2.3 Experiments and discussion [10 marks]

Experiment with the settings of M and ϵ. For example, what happens to classification accuracy as the number of components decreases? What about when the number of possible speakers, S, decreases? You will be marked on the detail with which you empirically answer these questions and whether you can devise one or more additional valid experiments of this type.

Additionally, your report should include short hypothetical answers to the following questions:

- How might you improve the classification accuracy of the Gaussian mixtures, without adding more training data?
- When would your classifier decide that a given test utterance comes from none of the trained speaker models, and how would your classifier come to this decision?
- Can you think of some alternative methods for doing speaker identification that don't use Gaussian mixtures?

---

3 Speech Recognition

Speech recognition is the task of correctly identifying a word sequence given an input speech sequence X. Typically this process involves language models, dictionaries, and grammars. In this section we will consider only a small subset of the acoustic modelling component. We will use the `Bayes Net Toolbox`, written for Matlab by Kevin Murphy, which is documented on-line at `http://bnt.googlecode.com`, along with some interface code that simplifies it for the case of continuous HMMs.

Continuous HMMs are exactly like discrete HMMs except that the observation probability matrix is replaced by the parameters of continuous probability density functions. This is necessary because the set of outputs is no longer discrete, but continuous. In this case, the outputs are all potential MFCC vectors. For our specific purposes, we will assume that each observation probability density is a Gaussian mixture model.

See Appendix A for important details on the file formats used.

3.1 Training and decoding Hidden Markov models [15 marks]

Using the interface functions `initHMM`, `loglikHMM`, and `trainHMM` in */u/cs401/A3_ASR/code*, write a simple scriipt, `myTrain`, that can be used to initialize and train continuous hidden Markov models for each phoneme in the data set. You should not modify the interface functions, nor submit them along with your assignment. Note that each model is trained on all data of a specific phoneme across *all* speakers; hence these models will be *speaker-independent*. `myTrain` must use the same HMM format as `initHMM` and `trainHMM`.

Once you have trained the models, write a script `myRun` that collects all phoneme sequences from the test data given their respective `*.phn` files. `myRun` must find the log likelihood of each phoneme sequence in the test data given each HMM phoneme model using the `loglikHMM` function. Report in your discussion on the proportion of correct identifications of the phoneme sequences in the test data.

3.2 Experiment and discussion [10 marks]

Experiment with changes to the parameters of this process, namely the number of mixtures per state, the number of states per sequence, and the amount of training data used. Your experiments should also include changes to the dimensionality of the data which can be performed here merely by considering the first d' dimensions of the mfcc data, where $d' < 14$. You should decide on what constitutes valid changes to these parameters, but your experiments should include at least two conditions for each of these four parameters, for a total of at least 16 experimental scenarios. Since the computational requirements increase as these parameters increase, you may consider experimenting with smaller values of these parameters.

Your discussion will be marked by the same general criteria as in Section 2.3.

3.3 Word-error rates [10 marks]

Imagine that we have now combined our phonetic HMMs into a larger speech recognition system that can recognize whole words and sentences. Complete the provided template (in `/u/cs401/A3_ASR/code/`) for `Levenshtein.m` in Matlab that computes the word-error rate between a test string and a reference string using Levenshtein distance. Assume that the cost of a substitution is 0 if the words are identical and 1 otherwise. The costs of insertion and deletion are both 1.

Assume that a speech recognition system produced the transcriptions for the *.wav files in the *Testing* directory and has written its hypotheses in the file *hypotheses.txt*, also in that directory. In that file, the i^{th} line is the hypothesis for `unkn_`i`.wav`.

Report on the word error rates between those transcriptions and the reference transcriptions (`*.txt`) according to the following measures for each utterance, and for the entire data set:

$$\begin{aligned} SE &= \text{proportion of substituted words} = \frac{\text{\# Substituted words}}{\text{\# Reference words}} \\ IE &= \text{proportion of inserted words} = \frac{\text{\# Inserted words}}{\text{\# Reference words}} \\ DE &= \text{proportion of deleted words} = \frac{\text{\# Deleted words}}{\text{\# Reference words}} \\ \text{proportion of total error} &= SE + IE + DE \end{aligned}$$

4 Actual Speech Recognition (and Synthesis (and Recognition))

Create a Matlab script, `ibmSpeech.m`, that recognizes the audio data in TIMIT, synthesizes the orthography of those data, recognizes *those* results, and compares the recognition results in both cases. Results will be stored in `discussion.txt`.

4.1 Original [5 marks]

First, start a **speech-to-text** service on BlueMix. Follow the tutorial at `https://www.ibm.com/smarterplanet/us/en/ibmwatson/developercloud/doc/speech-to-text/tutorial.shtml` to set it up. You will use calls equivalent to the following;

```
curl -u <username>:<password> -X POST
--header "Content-Type: audio/flac"
--header "Transfer-Encoding: chunked"
--data-binary @<path>0001.flac
"https://stream.watsonplatform.net/speech-to-text/api/v1/recognize?continuous=true"
```

For *each* utterance in the **Testing** directory, pass the associated `.flac` file[1] to BlueMix and retain the text in the returned transcript field[2]. Run your Levenshtein algorithm from section 3.3 on each returned transcript, comparing it to the known orthography in the associated `.txt` file of the utterance. For each utterance, report the recognized transcript and word error rate in `discussion.txt`.

4.2 Synthesized [10 marks]

Second, start a **text-to-speech** service on BlueMix. Follow `https://www.ibm.com/smarterplanet/us/en/ibmwatson/developercloud/text-to-speech/api/v1/#introduction` for more information. For each utterance in the **Testing** directory, decide whether the speaker is a (biological) male or female by using your results from section 2.2. I.e., if the guessed speaker's ID begins with an M, they are assumed to be male, otherwise they are assumed to be female. If they are assumed male, synthesize the associated `.txt` file with the *en-US_MichaelVoice* voice, otherwise use the *en-US_LisaVoice* voice. Be sure to specify in the `accept` field that you want output of type `audio/flac`.

Finally, for each synthesized flac that you obtain (one per utterance), pass *that* audio to the previous speech-to-text service, obtain the top recognized orthography, compute its WER with the Levenshtein code from section 3.3, and report each of these recognized orthographies and associated WER in `discussion.txt`.

Your `discussion.txt`, for section 4 should have orthographies and WER for each of the 30 original utterances, each of the 30 *synthesized* utterances, and a brief comparison (2-3 sentences at least) of the results across both sub-experiments.

[1] Watson on BlueMix accepts `.flac` audio files.

[2] If you obtain more than one alternative, retain the one with highest confidence.

5 Bonus [up to 10 marks]

We will give up to 10 bonus marks for innovative work going substantially beyond the minimal requirements. These marks can make up for marks lost in other sections of the assignment, but your overall mark for this assignment cannot exceed 100%.

You may decide to pursue any number of tasks of your own design related to this assignment, although you should consult with the instructor or the TA before embarking on such exploration. Certainly, the rest of the assignment takes higher priority. Some ideas:

Voice banking !!!!!!!!! What a wonderful time!

We are running a large study or 'normative' data in which people from the general population donate their speech (and language) data so that we can learn subtle differences in pathological populations. If you go to `https://www.cs.toronto.edu/talk2me/` (note that it's encrypted), you can obtain 1 bonus point per entire bonus session completed, ideally at least 1 day apart. There is no limit on your age, or first language. Currently, only the Chrome and most recent Firefox browser are supported. Create a new username for this assignment, and indicate your username in your submission.

Dimensionality reduction

Principal components analysis (PCA) is a method that converts some multivariate representation of data into a lower-dimensional representation by transforming the original data according to mutually orthogonal principal components. These principal components are defined by axes in the original data such that they have as high a variance as possible. By reducing the dimensionality of the data in this way, we can reduce the number of parameters necessary to model the data while being sensitive to original 'shape' of the data.

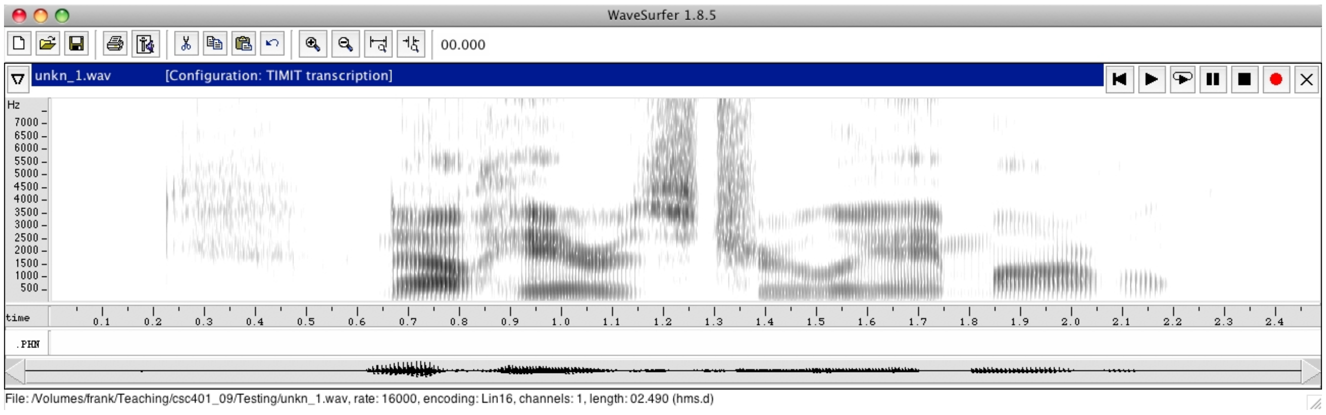
Look into principal components analysis and implement an algorithm that discovers a $d \times d'$ matrix W that transforms a d-dimensional vector, $\vec{x}$ into a d'-dimensional vector $\vec{y}$ through a linear transformation, where $d' < d$. Repeat a section of this assignment (i.e., speaker identification, or HMM training and testing) using data that has been transformed by principal components and report on what you observe through experimentation (e.g., for different values of d'). Submit all code and materials necessary to repeat your experiments (e.g., data or instructions).

Phoneme annotation

Try annotating a set of 10 sentence-level utterances (saved as `*.wav` files) at the phoneme level. You can record (and submit) these files yourself, or obtain them from some other source. You should use the WaveSurfer program, which can be obtained for Windows, Mac, and Linux here:
`http://www.speech.kth.se/wavesurfer/`. You will also need the CMU dictionary (see Introduction).
For each of the 10 wave files, perform the following:

1. Open the wave file in WaveSurfer. When given the option, select "TIMIT transcription" as your configuration. The result should look similar to Figure 1, with the editable ".PHN" pane.
2. Fully annotate all phoneme sections (including silence) in the wave file according to the CMU dictionary. The positions of the phoneme boundaries do not need to be exact, but should be reasonable.
3. Save the resulting transcription as `*.phn` files.

Submit all wave files and transcription files, along with a short discussion of your methodology.



File: /Volumes/frank/Teaching/csc401_09/Testing/unkn_1.wav, rate: 16000, encoding: Lin16, channels: 1, length: 02.490 (hms.d)

Figure 1: Example WaveSurfer window, including a spectrogram of `unkn_1.wav` .

Log likelihoods revisited

In previous sections, log-likelihoods were computed for continuous HMMs using the functions provided by the `Bayes Net Toolbox`. Write your own forward probability function without using that toolbox, assuming that you have a vecor `Pi` of initial probabilities, a matrix `A` of transition probabilities, and parameters `W`, `Mu`, and `Cov` for the Gaussian mixture models that govern the observations at each state.

The template file */u/cs401/A3_ASR/code/MYloglikHMM.m* contains an interface between the `Bayes Net Toolbox` and this simpler form of HMMs so that you can test your code. Add your code directly to this file and submit it.

6 General specification

We may test your code on different training and testing data in addition to those specified above. Where possible, do not hardwire directory names into your code. As part of grading your assignment, the grader may run your programs using test harness Matlab scripts. It is therefore important that your each of your programs precisely meets all the specifications and formatting requirements, including program arguments and file names.

If a program uses a file or helper script name that is specified within the program, it must read it either from the directory in which the program is being executed, or it must read it from a subdirectory of `/u/cs401` whose path is completely specified in the program. Do **not** hardwire the absolute address of your home directory within the program; the grader does not have access to this directory.

All your programs must contain adequate internal documentation to be clear to the graders. External documentation is not required.

6.1 Submission requirements

This assignment is submitted electronically. You should submit:

1. All your code for `gmmTrain.m`, `gmmClassify.m`, `myTrain.m`, `myRun.m`, `Levenshtein.m`, and `ibmSpeech.m` (including helper scripts, if any).
2. The files *unkn_*.lik*
3. Your discussion for all tasks, in a file called `discussion.txt`. Clearly demarcate the discussions of the three tasks — do not combine them into a single essay.
4. The `ID` file available from the course web-site.

The electronic submission must be made from the CDF submission site. Do not `tar` or `compress` your files, and do not place your files in subdirectories.

7 Using your own computer

If you want to do some or all of this assignment on your laptop or other computer, you will have to do the extra work of downloading and installing the requisite software and data. You take on the risk that your computer might not be adequate for the task. You are strongly advised to upload regular backups of your work to CDF, so that if your machine fails or proves to be inadequate, you can immediately continue working on the assignment at CDF. When you have completed the assignment, you should try your programs out on CDF to make sure that they run correctly there. **A submission that does not work on CDF will get zero marks.**

A Appendix: File formats

Each utterance is represented by the five following file types:

`*.wav`	The original speech waveform sampled at 16kHz.
`*.flac`	The waveform in flac format.
`*.mfcc`	The Mel-frequency cepstral coefficients obtained from an analysis of the waveform. Each line represents a 16ms frame of speech and consists of 14 space-separated floating point values.
`*.phn`	Sample-aligned phonetic transcription. Each line has the format [BEGIN] [END] [PHONE] where [BEGIN] and [END] are integers indexing the first and last samples of a span of the utterance with phonetic label [PHONE].
`*.wrd`	Sample-aligned word transcription. Each line has the format [BEGIN] [END] [WORD] where [BEGIN] and [END] are integers indexing the first and last samples of a span of the utterance with orthographic label [WORD].
`*.txt`	Orthographic transcription of the entire utterance. The line has the format [BEGIN] [END] [UTTERANCE] where [BEGIN] and [END] are the first and last samples of the utterance.

The MFCCs were obtained by applying filters to windows of 256 consecutive samples of the speech waveforms, so each frame represents $256/16000 = 0.016$ seconds of speech. These windows are moved in increments of 128 samples, so frame f represents a window of speech beginning at $0.008f$ seconds. Each MFCC frame consists of 13 cepstral coefficients, including the 0^{th} order coefficient, as well as log energy.

Note: Transcription files are in the TIMIT format. This format gives the starting and ending *samples* of each phoneme in the associated waveform file. Since the waveform is sampled at 16 kHz, and MFCC frames are windowed every 128 samples, you have to divide the numbers in these transcription files by 128 to match those in the `*.mfcc` files.

Note: The transcription files are 0-indexed (first frame is frame 0), whereas Matlab is 1-indexed (first dimension is 1). You may need to correct for this in your code.

Note: You can treat the /*h#*/ phoneme as a silence (/*sil*/).

B Appendix: Training Gaussian mixture models

Initialize θ

Initialize your model with valid probabilities.

Compute likelihood

Equation 4 is the observation probability for the m^{th} mixture component, given diagonal covariance matrices.

$$b_m\left(\vec{x}_t\right) = \frac{\exp\left[-\frac{1}{2}\sum_{n=1}^{d}\frac{\left(x_t[n]-\mu_m[n]\right)^2}{\Sigma_m[n]}\right]}{(2\pi)^{d/2}\sqrt{\prod_{n=1}^{d}\Sigma_m[n]}} \tag{4}$$

Given the feature vector $\vec{x}_t$ and parameters θ, the prior probability of the m^{th} Gaussian component is

$$p\left(m|\vec{x}_t;\theta\right) = \frac{\omega_m b_m\left(\vec{x}_t\right)}{\sum_{k=1}^{M}\omega_k b_k\left(\vec{x}_t\right)} \tag{5}$$

Update parameters

Given posterior probabilities computed using Equations 4 and 5, we want to update estimates for the mixture weights, means, and diagonal covariance matrices. These are accomplished thus:

$$\begin{aligned}\hat{\omega}_m &= \frac{\sum_{t=1}^{T} p\left(m|\vec{x}_t;\theta\right)}{T}\\ \hat{\vec{\mu}}_m &= \frac{\sum_{t=1}^{T} p\left(m|\vec{x}_t;\theta\right)\vec{x}_t}{\sum_{t=1}^{T} p\left(m|\vec{x}_t;\theta\right)}\\ \hat{\Sigma}_m &= \frac{\sum_{t=1}^{T} p\left(m|\vec{x}_t;\theta\right)\vec{x}_t^2}{\sum_{t=1}^{T} p\left(m|\vec{x}_t;\theta\right)} - \hat{\vec{\mu}}_m^2\end{aligned} \tag{6}$$

In the third equation, the square of a vector on the right-hand side is defined as the component-wise square of each dimension in the vector.

C Appendix: TIMIT phone set

Tables 1, 2, and 3 list the permissible TIMIT phone symbols. Note the addition of plosive closure phonemes (e.g., /bcl/, /dcl/). These phonemes are essentially silence and precede their associated plosive phoneme.

	symbol	**example word**	**example transcription**
stops			
	b	bee	BCL B iy
	d	dumb	DCL D ah m
	g	gum	GCL G ah m
	p	pea	PCL P iy
	t	tea	TCL T iy
	k	key	KCL K iy
	dx	muddy, dirty	m ah DX iy, dcl d er DX iy
	q	bat	bcl b ae Q
affricates			
	jh	joke	DCL JH ow kcl k
	ch	choke	TCL CH ow kcl k
fricatives			
	s	sea	S iy
	sh	she	SH iy
	z	zone	Z ow n
	zh	azure	ae ZH er
	f	fin	F ih n
	th	thin	TH ih n
	v	van	V ae n
	dh	then	DH e n
nasals			
	m	mom	M aa M
	n	noon	N uw N
	ng	sing	s ih NG
	em	bottom	b aa tcl t EM
	en	button	b ah q EN
	eng	washington	w aa sh ENG tcl t ax n
	nx	winner	w ih NX axr
semivowels and glides			
	l	lay	L ey
	r	ray	R ey
	w	way	W ey
	y	yacht	Y aa tcl t
	hh	hay	HH ey
	hv	ahead	ax HV eh dcl d
	el	bottle	bcl b aa tcl t EL

Table 1: TIMIT consonant phone-set.

	symbol	example word	example transcription
vowels			
	iy	beet	bcl b IY tcl t
	ih	bit	bcl b IH tcl t
	eh	bet	bcl b EH tcl t
	ey	bait	bcl b EY tcl t
	ae	bat	bcl b AE tcl t
	aa	bott	bcl b AA tcl t
	aw	bout	bcl b AW tcl t
	ay	bite	bcl b AY tcl t
	ah	but	bcl b AH tcl t
	ao	bought	bcl b AO tcl t
	oy	boy	bcl b OY
	ow	boat	bcl b OW tcl t
	uh	book	bcl b UH kcl k
	uw	boot	bcl b UW tcl t
	ux	toot	tcl t UX tcl t
	er	bird	bcl b ER dcl d
	ax	about	AX bcl b aw tcl t
	ix	debit	dcl d eh bcl b IX tcl t
	axr	butter	bcl b ah dx AXR
	ax-h	suspect	s AX-H s pcl p eh kcl k tcl t

Table 2: TIMIT vowel phone-set.

symbol	description
pau	pause
epi	epenthetic silence
h#	begin/end marker (non-speech events)
1	primary stress marker
2	secondary stress marker

Table 3: Other TIMIT codes.