Slide 1

Training Hidden Markov Models using Haskell

David M. Goblirsch

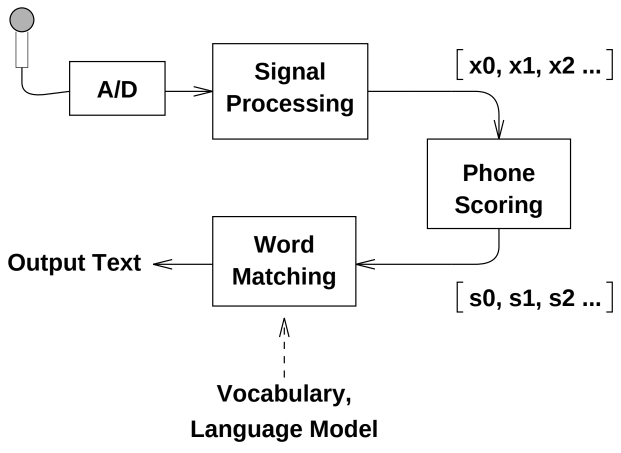
Dagstuhl Seminar on
Functional Programming in the Real World
May 1994

In this talk I will describe one approach for training hidden Markov models (HMMs) for automatic speech recognition. This approach has been implemented as a suite of programs, some written in Haskell. Others were prototyped in Haskell, but had to be rewritten in C in order to improve efficiency.

This talk will concentrate on a high level description of the algorithm. The implementation details are in the accompanying report.

Slide 2

Speech Recognition System Modules



Slide 3

A Statistical Approach to Speech Recognition

- Approach: Given a finite vocabulary $\mathcal{V}$ and a sequence of acoustic measurements $[x_1, \ldots, x_N]$, find the sequence of words $[w_1, \ldots, w_L]$ from $\mathcal{V}^*$ that maximizes

$$\underbrace{p[x_1, x_2, \ldots, x_N \mid w_1, \ldots, w_L]}_{\text{acoustic model}} \cdot \underbrace{P[w_1, \ldots, w_L]}_{\text{language model}}$$

- Run-time vs. design-time system
- Our problem: develop the acoustic model

The acoustic measurements are vectors of real numbers. They will be described in a little more detail on Slide 8. If the dimension of each vector is n, then the expression

$$p[x_1, x_2, \ldots, x_N \mid w_1, \ldots, w_L]$$

is a conditional probability density function over the N-fold cartesian product of copies of $\mathcal{R}^n$. On the other hand, the words are drawn from a finite alphabet, $\mathcal{V}$, so the expression

$$P[w_1, \ldots, w_L]$$

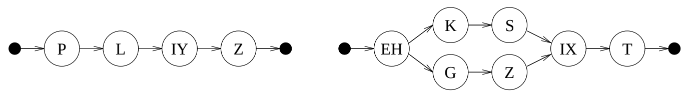
is a discrete probability distribution over the (countable) space $\mathcal{V}^*$.

This talk is about one way of implementing the acoustic model and about the Haskell programs we've written as part of a suite of programs for estimating the model parameters.

Slide 4

Phonetic Modeling

- Words are modeled as networks of basic sound units called *phones*



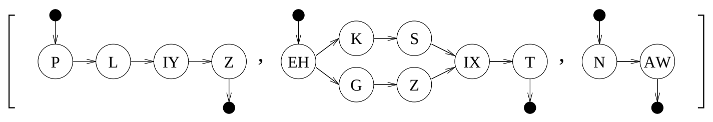
- North American English requires about 46 phones
- Sentences are modeled as networks by concatenating the individual word networks, with optional interword networks for pauses and other noises

Slide 5

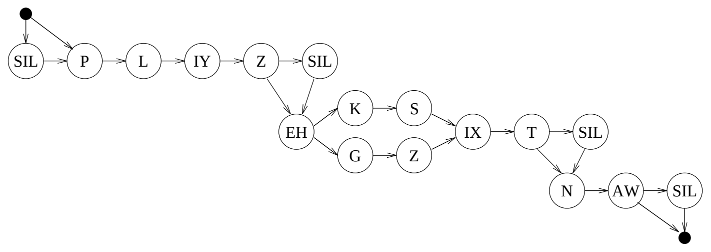
Modeling Sentences

"PLEASE EXIT NOW"

$\Downarrow$



$\Downarrow$



The text is split into individual words. Then, the pronunciation model for each word is retrieved from a dictionary. Finally, the models are joined together into a *pronunciation network*, with optional silence models (or, more generally, any between-word sound models) placed at the beginning of the sentence and after each word.

Slide 6

Connecting Phones to the Acoustic Measurements

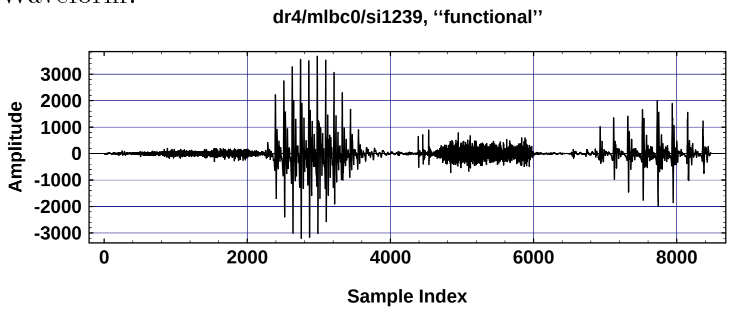
- We need to develop a probability model for each phone q

$$p[x_{i+1}, \ldots, x_{i+K} \mid q], \qquad K_{\min}^{(q)} \leq K \leq K_{\max}^{(q)}$$

Slide 7

Example

- Waveform:



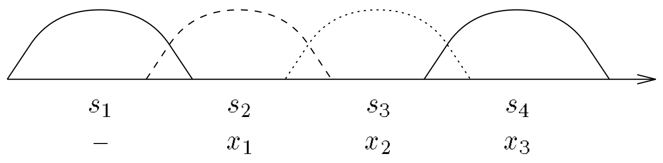
- Dictionary transcription: /f ah ng k sh ix n el/

This example waveform from the TIMIT database, an American male saying the word "functional," illustrates the time-varying nature of speech.

Slide 8

The Analysis of Speech

- Short-time Spectral Analysis



- Typical window width: 20ms Typical offset: 10ms
- Typically, the feature vector dimension lies between 18 and 30

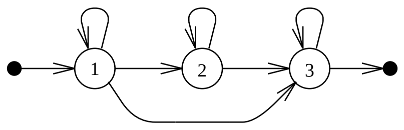
A number of different spectral representations have been proposed by different researchers. The details of how Haskell can be used to implement one standard approach to performing this "short-time" spectral analysis is being written up and will be submitted to the *Journal of Functional Programming.* Currently, we use a C program written at the start of our research. This program is part of the run-time system, so it needs to run in real time.

The dimension of the feature vectors depends on the nature of the spectral parameters.

Slide 9

Hidden Markov Models

- Models the switching of probability laws by a Markov chain, but the chain states are hidden from the observer



Initial Probabilities	$\pi_1,\ \pi_2,\ \pi_3$
Exit Probabilities	$\tau_1,\ \tau_2,\ \tau_3$
Transition Probabilities	$a_{11},\ a_{12},\ \ldots,\ a_{33}$
Density functions	$g_1(x),\ g_2(x),\ g_3(x)$

Slide 10

Gaussian Mixture Densities

- Multivariate Gaussian Density

$$\mathcal{N}(x \mid \mu, \Sigma) = \frac{1}{(2\pi)^{n/2}(\det \Sigma)^{1/2}} \exp\{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\}$$

- Mixture Densities: Given density functions $f_1, \ldots, f_M$ and positive real numbers $c_1, \ldots, c_M$ for which

$$\sum_{i=1}^{M} c_i \; = \; 1,$$

let

$$g(x) = \sum_{i=1}^{M} c_i f_i(x).$$

Slide 11

Gaussian Mixture Densities: Diagonal Covariance

- If each Gaussian component density is assumed to have a diagonal covariance matrix, then

$$\mathcal{N}(x \mid \mu, \Sigma) = \frac{1}{(2\pi)^{N/2}\prod_{j=1}^{N}\sigma_j} \exp\{-\frac{1}{2}\sum_{j=1}^{N}(x_j-\mu_j)^2/\sigma_j^2\}$$

Evaluating these densities is a major computational burden. In the first version of the software, the code for evaluating the exponential portion of the density formula for a single Gaussian density was implemented this way:

```
eval_comp_exp  xs  mu  sigma =
  let  exponent = sum [ ((x-m)/s)^2 | (x,m,s) <- zip3 xs mu sigma]
  in   exp (-exponent / 2.0)
```

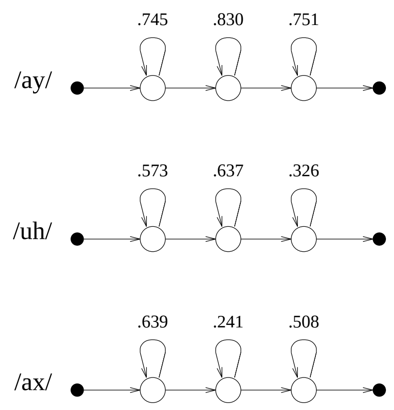
Later, it was revised to:

```
eval_comp_exp  xs mu sigma =
  exp (-0.50 * eval_exponent xs mu sigma)

eval_exponent (x:rxs) (u:rus) (s:rss) =
  let  t = (x-u)/s
  in   t*t  + eval_exponent rxs rus rss

eval_exponent [] [] [] = 0.0
```

This simple change reduced the overall execution time by 55%!

Slide 12

Hidden Markov Models: Examples



The values for these models were calculated using our suite of programs. The training set consisted of 3333 sentences drawn from the Air Travel Information System (ATIS) corpus and the Resource Management (RM) corpus, both distributed on CD ROM by the Linguistic Data Consortium.

The phone /ay/ is the vowel sound in words such as “like” and “high.” It is a dipthong. On average, it lasts longer than the other two phones, a fact that is captured by the higher probabilities assigned to its self-loops. The phone /uh/ is the vowel sound in words such as “look” and “book.” The phone /ax/, called *schwa* and commonly represented as an upside-down ‘e’ in dictionaries, is an unstressed neutral vowel, appearing in words such as “the” and in unstressed syllables.

Slide 13

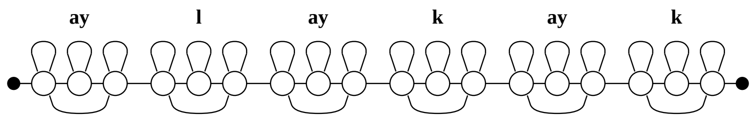
Modeling Sentences

"I like Ike."

$\Downarrow$

[ay l ay k ay k]

$\Downarrow$



Slide 14

HMMs: Three Basic Problems

1. Compute $p[x_1, x_2, \ldots, x_N|\lambda]$ efficiently
   - The Forward algorithm
   - The Backward algorithm
2. Given $[x_1, x_2, \ldots, x_N]$ and λ, estimate $[q_1, q_2, \ldots, q_N]$
   - The Viterbi algorithm
3. Estimate the parameters of λ
   - Baum-Welch (Expectation-Modification)
   - Segmental K-Means

We will talk in detail about the *state sequence estimation* problem and the Viterbi algorithm used to solve it, and give an overview of the Segmental K-means algorithm.

Slide 15

State Sequence Estimation

- One approach:

$$\max_{q_1,\ldots,q_N} p[q_1,\ldots,q_N,x_1,\ldots,x_N \mid \lambda]$$

- Example of a joint probability/density calculation for $N = 4$:

$$\pi_{q_1} g_{q_1}(x_1) a_{q_1 q_2} g_{q_2}(x_2) a_{q_2 q_3} g_{q_3}(x_3) a_{q_3 q_4} g_{q_4}(x_4)$$

Slide 16

Deriving the Viterbi Algorithm

Let $i, j \in \{1, 2, \ldots, S\}$. Define

$$\delta_t[j] = \max_{q_1,\ldots,q_{t-1}} \pi_{q_1} g_{q_1}(x_1) a_{q_1 q_2} g_{q_2}(x_2) \cdots a_{q_{t-1} j} g_j(x_t)$$

Then

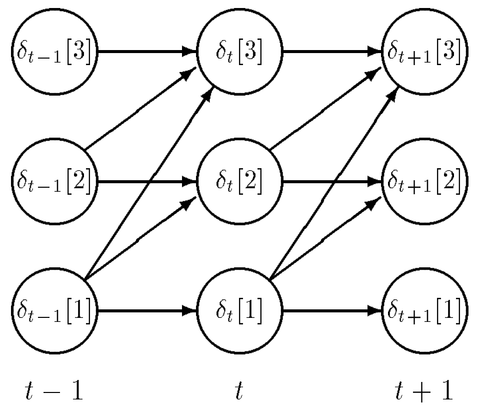
$$\delta_{t-1}[i] = \max_{q_1,\ldots,q_{t-2}} \pi_{q_1} g_{q_1}(x_1) a_{q_1 q_2} g_{q_2}(x_2) \cdots a_{q_{t-2} i} g_i(x_{t-1})$$

This leads to the recursion formula

$$\delta_t[j] = \max_i \{\delta_{t-1}[i] a_{ij}\} \cdot g_j(x_t)$$

Slide 17

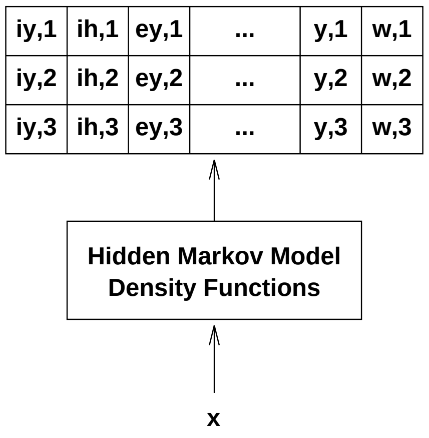
The Viterbi Trellis



$$\delta_t[j] = \max_i \{\delta_{t-1}[i] a_{ij}\} \cdot g_j(x_t)$$

Slide 18

Hidden Markov Models: Phone Scoring



Slide 19

The Viterbi Algorithm

- Initialization: $\delta_1[i] = \pi_i g_i(x_1)$
- Recursion: $$\begin{aligned} \delta_t[j] &= \max_i\{\delta_{t-1}[i]a_{ij}\} \cdot g_j(x_t) \\ \psi_t[j] &= \arg\max_i\{\delta_{t-1}[i]a_{ij}\} \end{aligned}$$
- Termination: $$\begin{aligned} P^* &= \max_i\{\delta_N[i]\} \\ q_N^* &= \arg\max_i\{\delta_N[i]\} \end{aligned}$$
- Backtrace: $q_{t-1}^* = \psi_t[q_t^*]$

The indices i and j range over the integers from 1 through S. S is the number of HMM states. ψ is the *backtrace array.* The recursion variable t goes from 2 to N (recall that N is the number of observation vectors).

Slide 20

Preparing a Training Database

- Record many different speakers
  - Create a database of waveform and transcription files.
- Process each waveform file to produce a feature vector file (C program).
- Convert each training text to a phonetic pronunciation model ("`BatchTranscribe`")

Slide 21

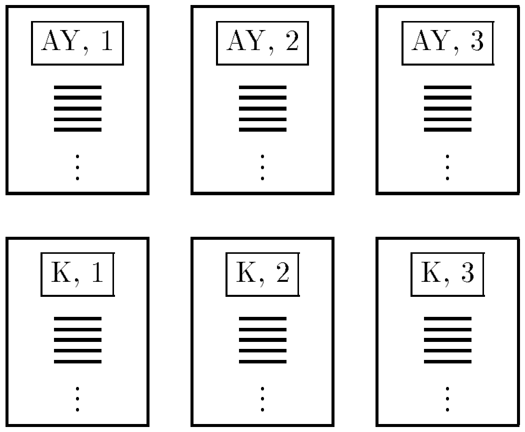
Segmental K-means Training Algorithm

1. Convert the phonetic pronunciation network for each training utterance into an HMM and perform the Viterbi alignment algorithm (“`BatchAlign`”)
2. Count state transitions to estimate new HMM transition probabilities (“`ComputeNewDgs`”)
3. Group feature vectors by phone and state (C program)
4. Use vector quantization and the EM algorithm to get observation density parameters (C program, ESPS)
5. Repeat steps 1 through 4 until convergence (Shell script)

ESPS is a commercial package of speech processing tools.

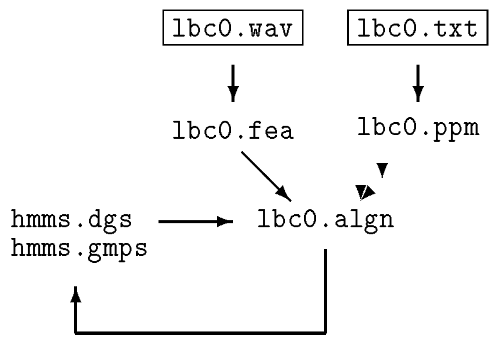
Slide 22

Mixture Density Training Data



Slide 23

Flow of Processing



Slide 24

The System Software (Haskell Modules)

- `Phones`
- `Pronunciations`
- `HmmDigraphs`
- `HmmDensities`
- `Viterbi`
- `HmmConstants`
- `Alignments`

Slide 25

The System Software (Haskell Programs)

- `ConvertLinearDic`
- `Transcribe`
- `BatchTranscribe`
- `BatchAlign`
- `ComputeNewDgs`
- `CountAlignmentDiffs`

Slide 26

Haskell Strengths

- Shorter code
  - Polymorphism enabled us to retain the “shape” of a data structure but with different data values.
  - Higher-order functions
- Literate programming

Slide 27

Problems with Haskell

- Need a "Native" module for reading/writing data in the machine's native format.
- Need automatic program transformations
- Troubles with lazy evaluation and UNIX restrictions on the number of files a program can have open at one time.
- Speed

Slide 28

References

- L. R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition." *Proc. of the IEEE*, Vol. 77, No. 2, Feb. 1989, 257–286.
- L. R. Rabiner and B.-H. Juang. *Fundamentals of Speech Recognition.* PTR Prentice-Hall, 1993.