Speech Processing, Recognition and Understanding Overview

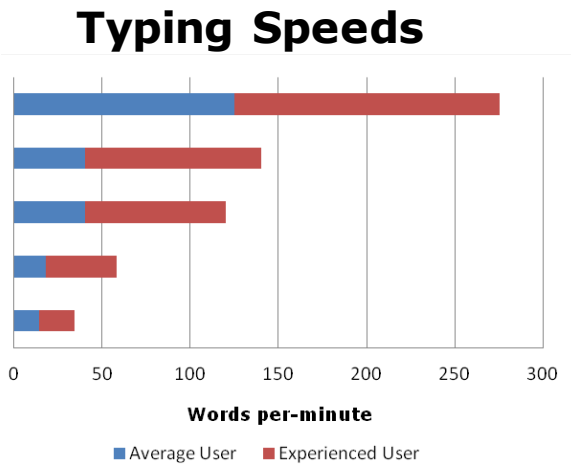
Speech .*

- Speech Processing: General, e.g. find out if audio is loud
- Speech Recognition: Usually refers to speech to text conversion
- Speech Understanding: Too hard. Often used to denote that we “understand” what action to take based on speech. e.g. “Play stairway to heaven” will result in song playing
- Communication understanding: VERY loaded. Is the other person happy, negative, sarcastic, suicidal,…. too loaded! (This is my lab’s focus)

Speech vs Text vs "understanding"

- Speech is the most natural and powerful form of communication between humans
  - Natural: No additional learning required
  - Flexible: Adapt to dialogue partners, environment and situations
  - Efficient: Communicate large amounts of information
  - Informative: about speaker, their cognitive-state, environment
- Speech vs text vs meaning
  - There is a lot of "what's in speech that is not in text"
  - Text is just a convenient medium to write things (e.g. references) down

With speech, interaction becomes independent of screen estate



If accurately recognized, speech is three times faster than QWERTY (Basapur et al. '07)

Only plausible interaction modality for 800 million non-literate users

Speed of "communication"

- In characters per minute

Mode	Standard	Best
Handwriting	200	500
Typewriter	200	1.000
Stenography	500	2.000
Speech	1.000	4.000

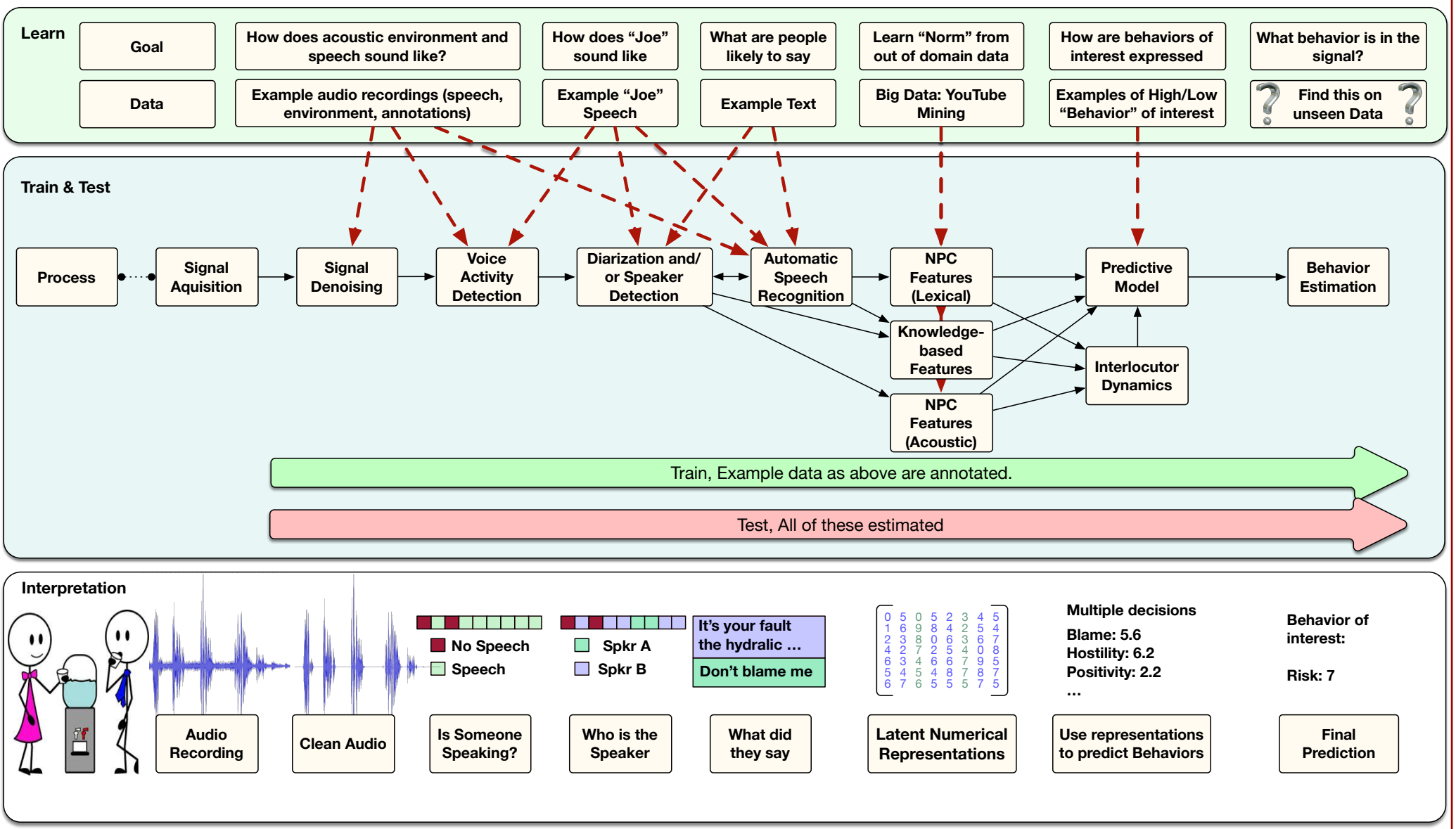
Speech Pipeline Overview

(Within the context of my research…)

USC Viterbi School of Engineering

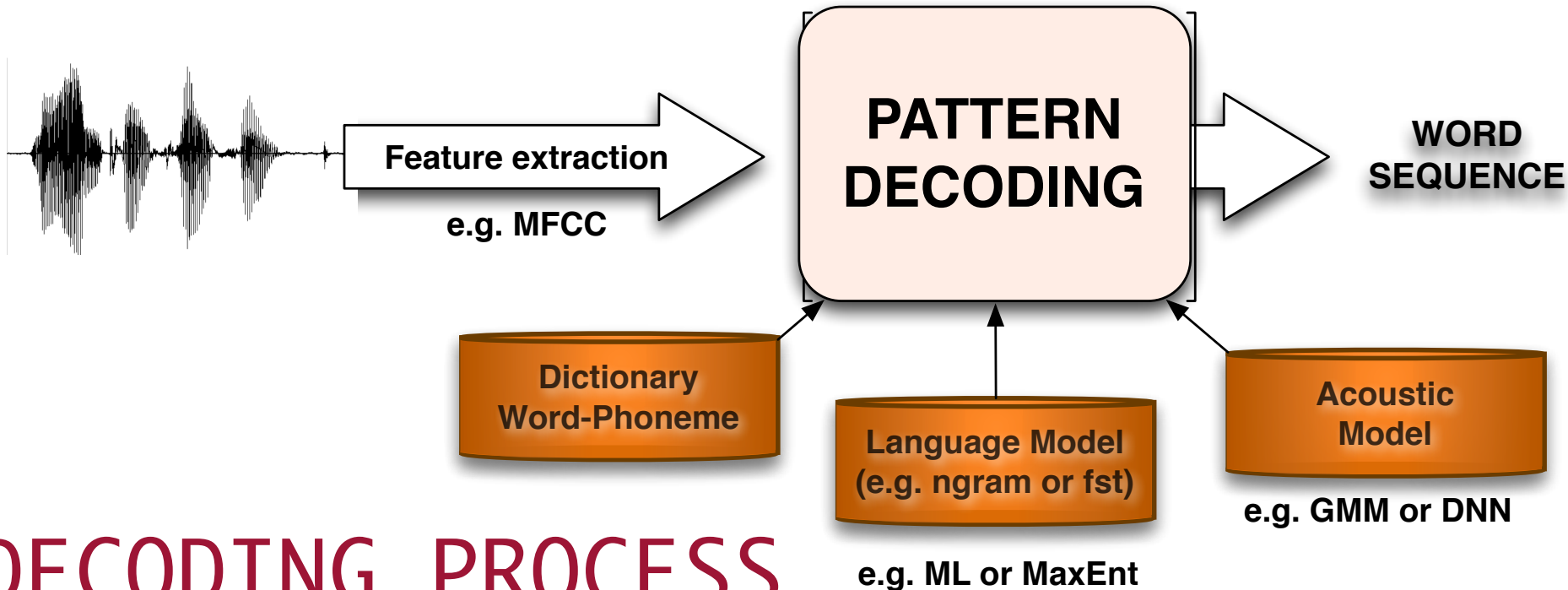
MY LAB AND PART OF CLASS OVERVIEW

SCUBA



Automatic Speech Recognition Overview

Automatic Speech Recognition (ASR) Overview



DECODING PROCESS

- MFCC: mel frequency cepstral coefficient
- HMM: Hidden Markov Model
- DNN: Deep Neural Network
- N Gram: Probabilistic word sequence model with history dependence on the past N-1 words

State of the Art in Speech Recognition

- LVCSR = Large Vocabulary Speech Recognition
- ASR = Automatic Speech Recognition
- WER = Word Error Rate (can be above 100%)
- Current state of the art error rates range dramatically by task:

Task	Vocabulary	WER
Digits	11	0.5
Read speech	5K	3
Read speech	20K	3
Broadcast	64K	10
Conversational	64K	20

These are close-

What is our vocabulary?

ASR Training

To understand ASR, understand how it gets trained

ASR Training Process

Separate speech from background first

Feature extraction

feature vector | feature vector | feature vector | feature vector | feature vector

Transcript of above

Dictionary Word-Phoneme

Phonetic transcription

Mostly human-made, especially in non-phonetic languages like English

Training (e.g. Baum Welch)

Models

TRAINING PROCESS

Gaussian Mixture Acoustic Model

Millions of words of representative transcripts for the domain

Feature extraction

Language Model (e.g. ngram)

Speech Recognition (Decoding) Process

- Decoding:
  - Recognized Word sequence = the word sequence that is most probable given the observations

$$\hat{W} = \arg\max_{W \in D} P(W|O)$$

  - It is mathematically the same as (Bayes rule)

$$\hat{W} = \arg\max_{W \in D} \frac{P(O|W)P(W)}{P(O)}$$

  - And we can drop the common denominator

$$\hat{W} = \arg\max_{W \in D} P(O|W)P(W)$$

Acoustic Model

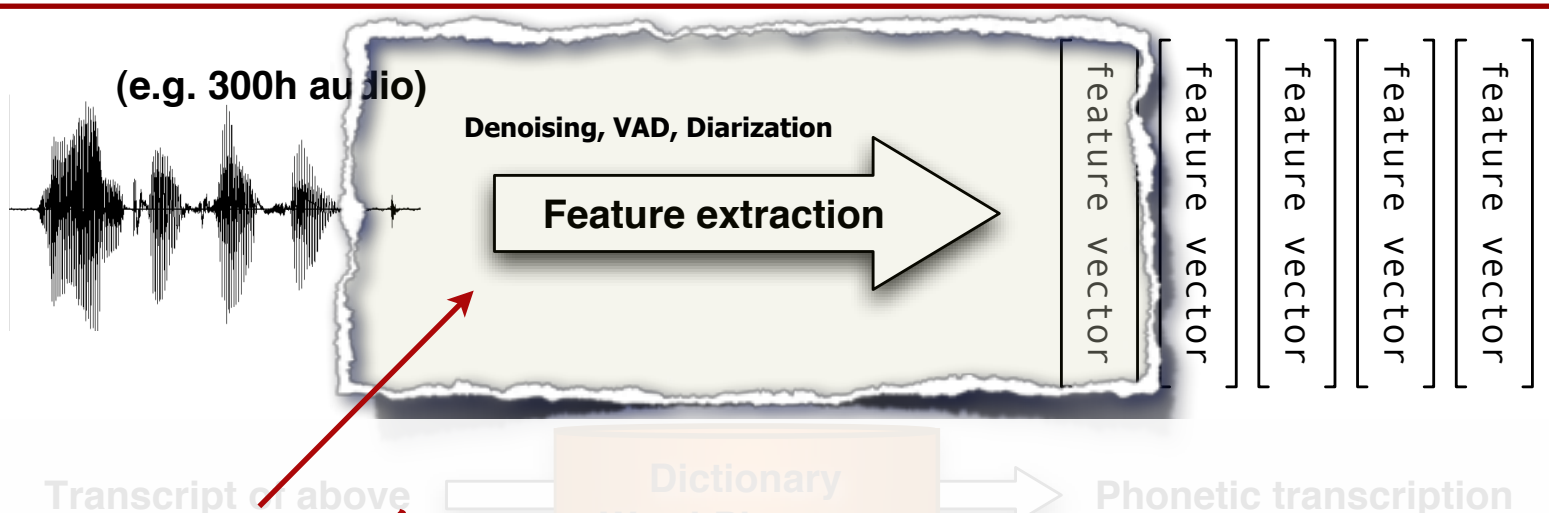
Language Model

  - Real life:

$$\hat{W} = \arg\max_{W \in D} P(O|W)P(W)^{N}$$

Bias weight

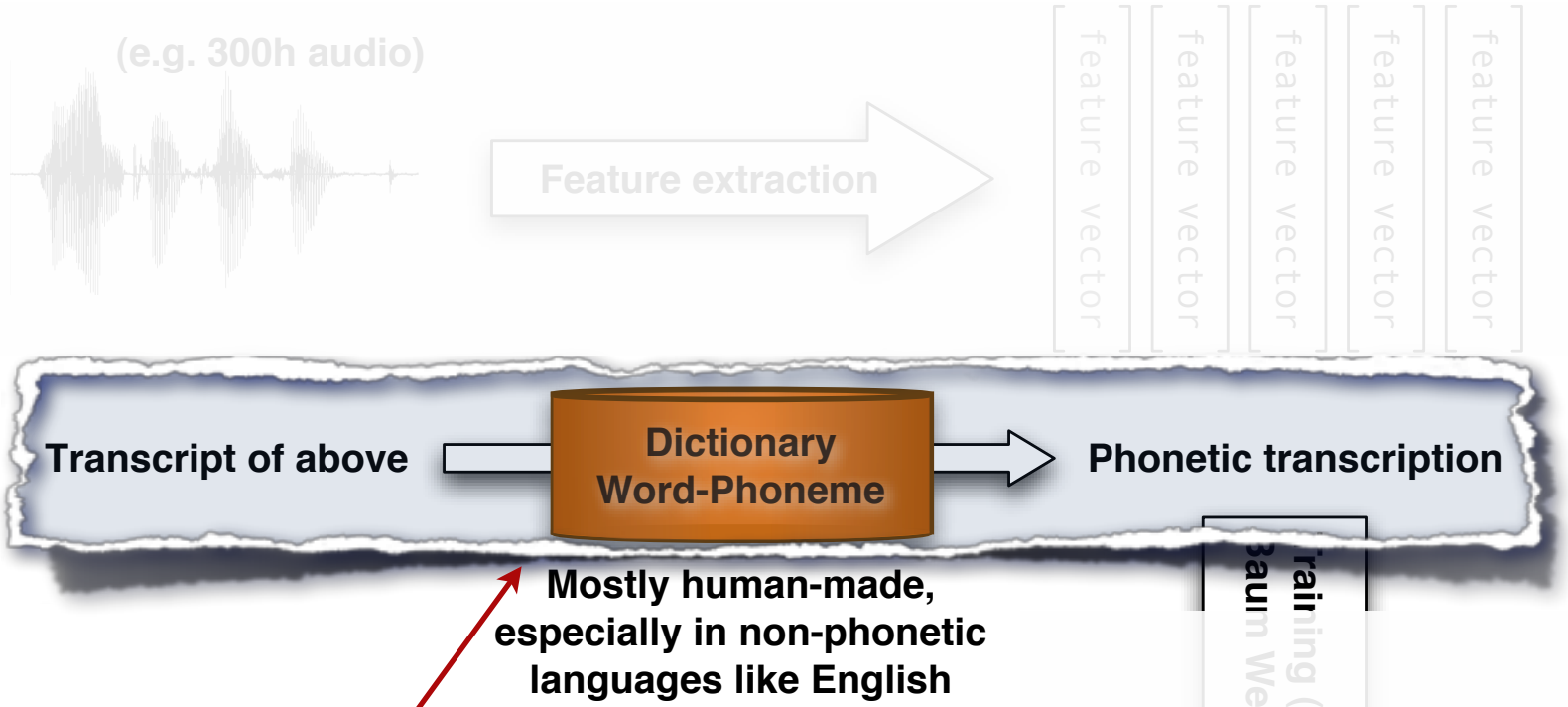
ASR Training: Feature Extraction



Feature extraction: Relates to:

- Denoising
- Robust features for
  - speech recognition
  - diarization
  - Speech (or Voice) Activity Detection
- etc

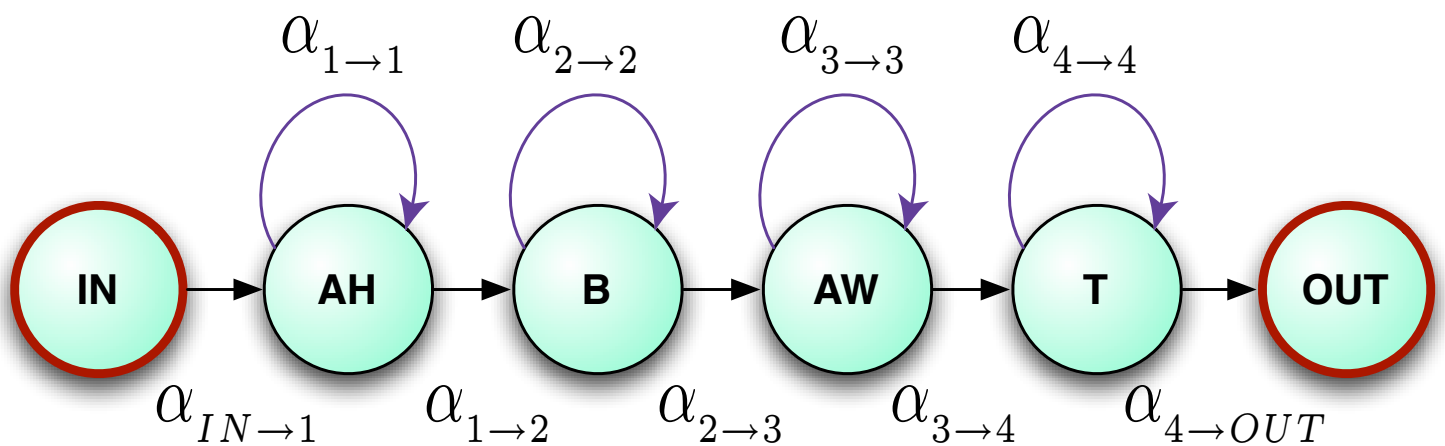
ASR Training: Dictionary

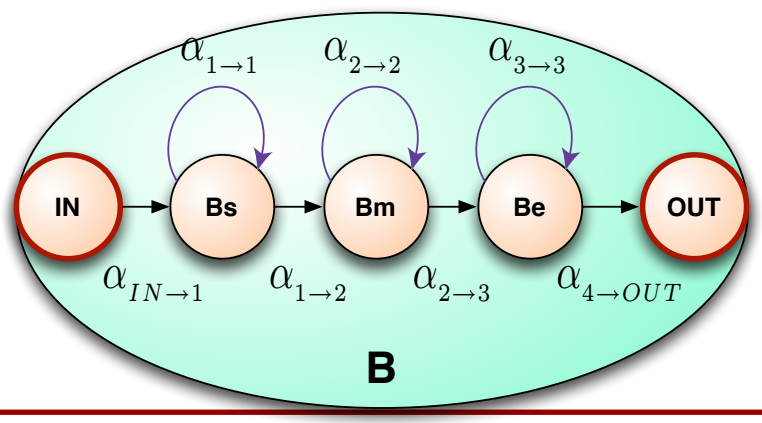


ASR Training: Dictionary

- In simple representation:
  - ABOUT AH B AW T
  - ABSORPTION AH B S AO R P SH AH N
  - ABSORPTION(2) AH B **Z** AO R P SH AH N

- But in reality each of these are a Hidden Markov Model state:





ASR Training: Dictionary Internally

In reality it is more complicated

- We use triphone models
  - –ABOUT _AH$_{B\ AH}$BAW$_{B\ AW}$AWT$_{AW}$T_
  - –ABSORPTION _AH$_{B\ AH}$BS$_{B\ S}$SAO$_{S\ AO}$AOR$_{AO\ R}$RP$_{R\ P}$PSH$_{P\ SH}$SHAH$_{SH\ AH}$AHN$_{AH\ N}$N_

⇒For a phoneme set of 50 phonemes (~English)

potentially 50^3 Triphones of 3 states each

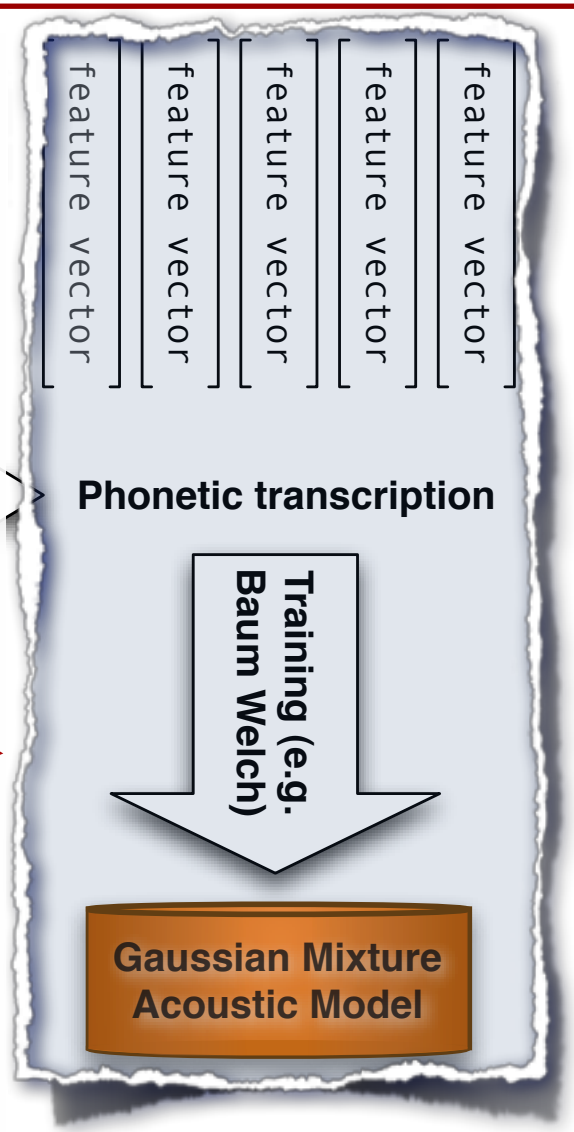
- Reduce space through ‘tying’ states (say down to 10K states)

- Every word in the dictionary is represented by a Hidden Markov Model based on these states

ASR Training: Acoustic Model

- Gives a prior on what speech sounds like
- Needs to match the domain where ASR will be used

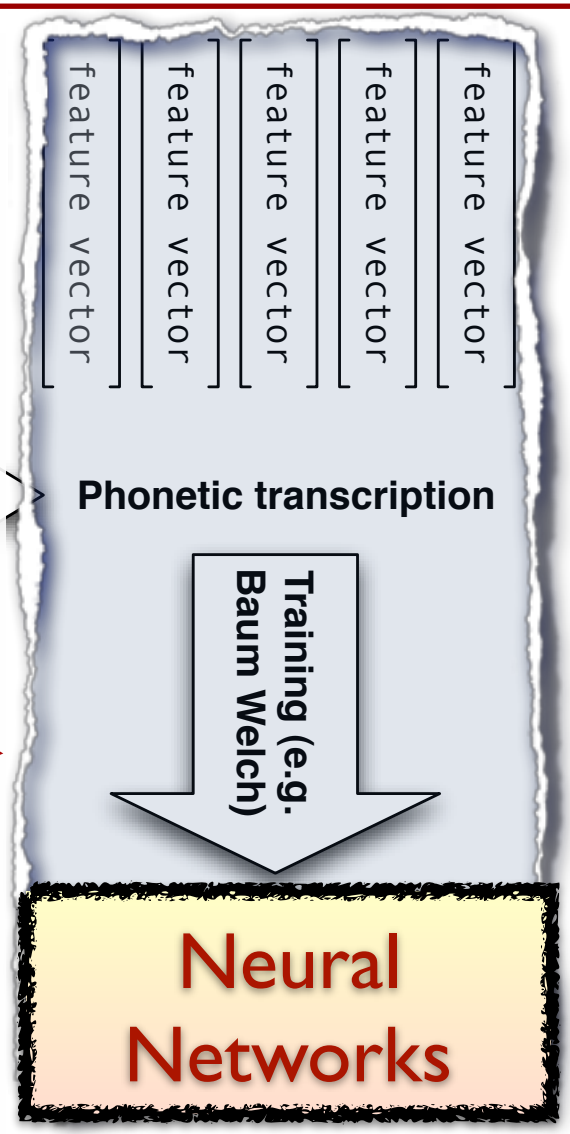
- Usual problem: training data is mismatched with domain data



ASR Training: Acoustic Model

- Gives a prior on what speech sounds like
- Needs to match the domain where ASR will be used

- Usual problem: training data is mismatched with domain data



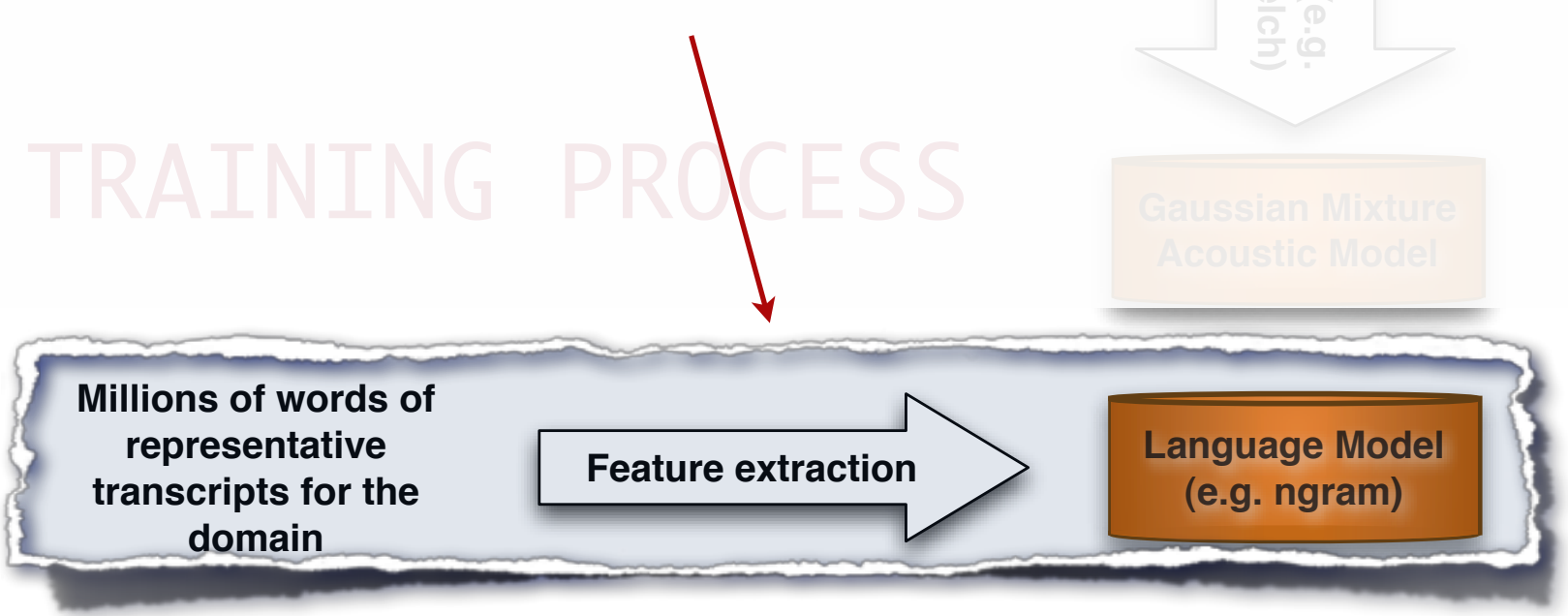
ASR Training: Language Model

Gives a prior on what people are likely to say

Every domain has different priors

Need lots and lots of data (millions of words)

Data starved in most domains



ASR Training: Language Model

- Second term:

$$\hat{W} = \arg\max_{W \in D} P(O|W) P(W)^N$$

Acoustic Model

Language Model

- P(W) can be extracted from existing text:

$$P(W) = P(W_1, W_2, ...W_n) = P(W_1)P(W_2|W_1)P(W_3|W_1W_2....)P(W_n|W_1W_2...W_{n-1})$$

- For simplicity and feasibility approximate with:

$$P(W) = P(W_1, W_2, ...W_n) = P(W_1)....P(W_{n-1}|W_{n-3}W_{n-2})P(W_n|W_{n-2}W_{n-1})$$

- When we don't have enough data - next best:

$$p(w_3|w_1, w_2) = \begin{cases} \text{if(trigram exists)} & P_3(w_1, w_2, w_3) \\ \text{else if(bigram } w_1, w_2 \text{ exists)} & BOW(w_1, w_2)P(w_3|w_2) \\ \text{else} & P(w_3|w_2) \end{cases}$$

Domain complications - Topics (Motivational Interviewing for addiction)

Example domain:
Topics LM:
Pretty good data…
but still very small!
(Compare with 20bil parameters in the general Google LM)

One 'FAULT' unigram:

```
-4.259933       fault
```

Results in 30 bigrams

```
-3.086692       my fault
```

and 2 trigrams

```
-1.545041       was my fault
```

Domain: Really data starved. Very few potential n-grams seen, especially 2+ grams

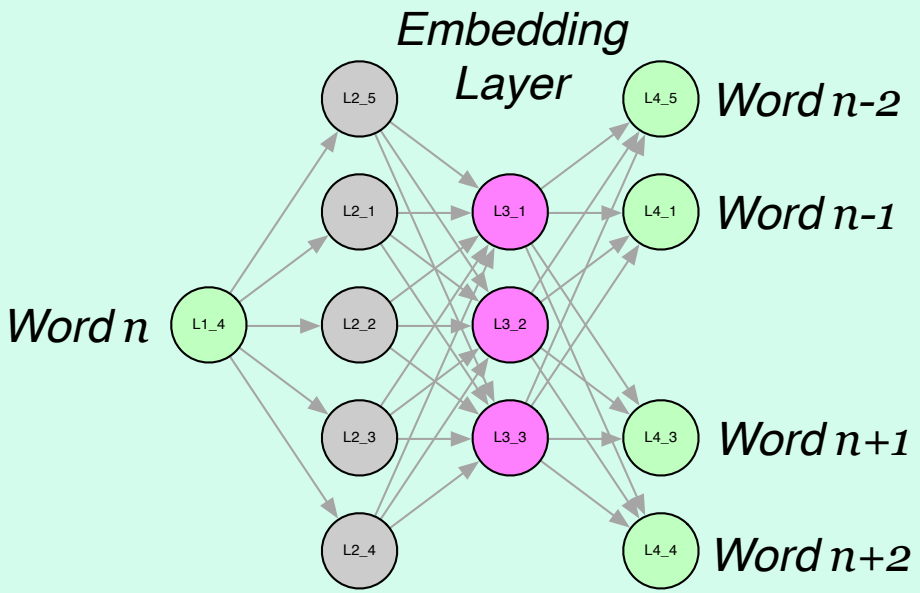
```
ngram 1=11494
ngram 2=173873
ngram 3=120996
```

Our data is a drop in the bucket!!

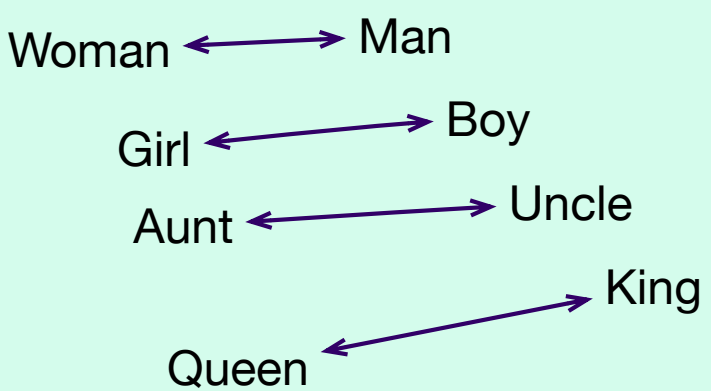
RNN LM's

- Enable Out-of-Domain Knowledge Transfer

Train on Out of Domain: Word2Vec Embedding

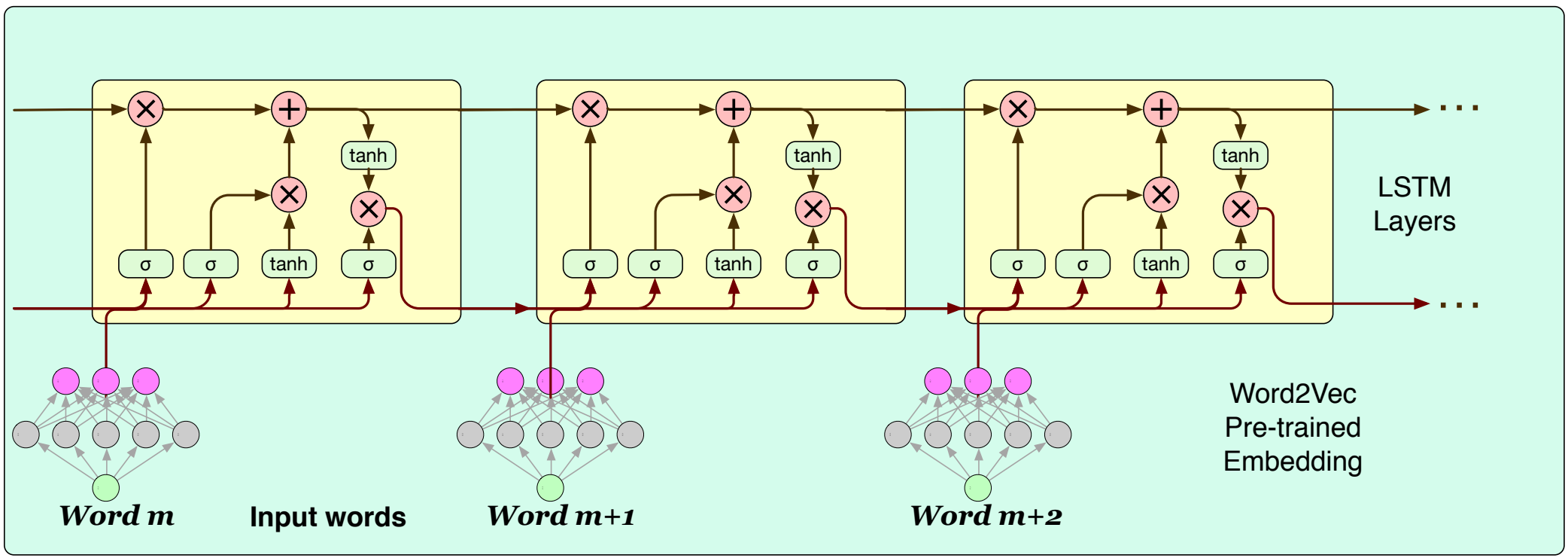


Learns Associations:



RNN LM's… work in progress

- RNN LM's can provide out of domain knowledge
- Still issues in integrating with lattices…



ASR Decoding Process

- Decoding:

$$\hat{W} = \arg\max_{W \in D} P(O|W)P(W)^N$$

- Every frame:
  - Birth of new words: this is probabilistic so hundreds of words are potentially starting every 10ms
  - Lexical Tree like search makes this faster (i.e. If we have seen phonemes X Y then all the words starting from X Y will be searched, but not remaining words)
  - As we move forward we can prune paths based on:
    - Maximum total alive words at any time instant
    - Maximum new words at any time instant
    - Pruning low probability paths by deeming them un-viable
    - Constraining total search space (dangerous), etc
  - Pruning reduces performance, so a good LM, and AM reduces the probability of pruning good paths

- Real time systems, “bad” LM, large/mismatched domains

Summary

- ASR aspects
  - Needed:
    - Representative audio
    - Transcriptions of the audio
    - Large amounts of representative text (in the millions)

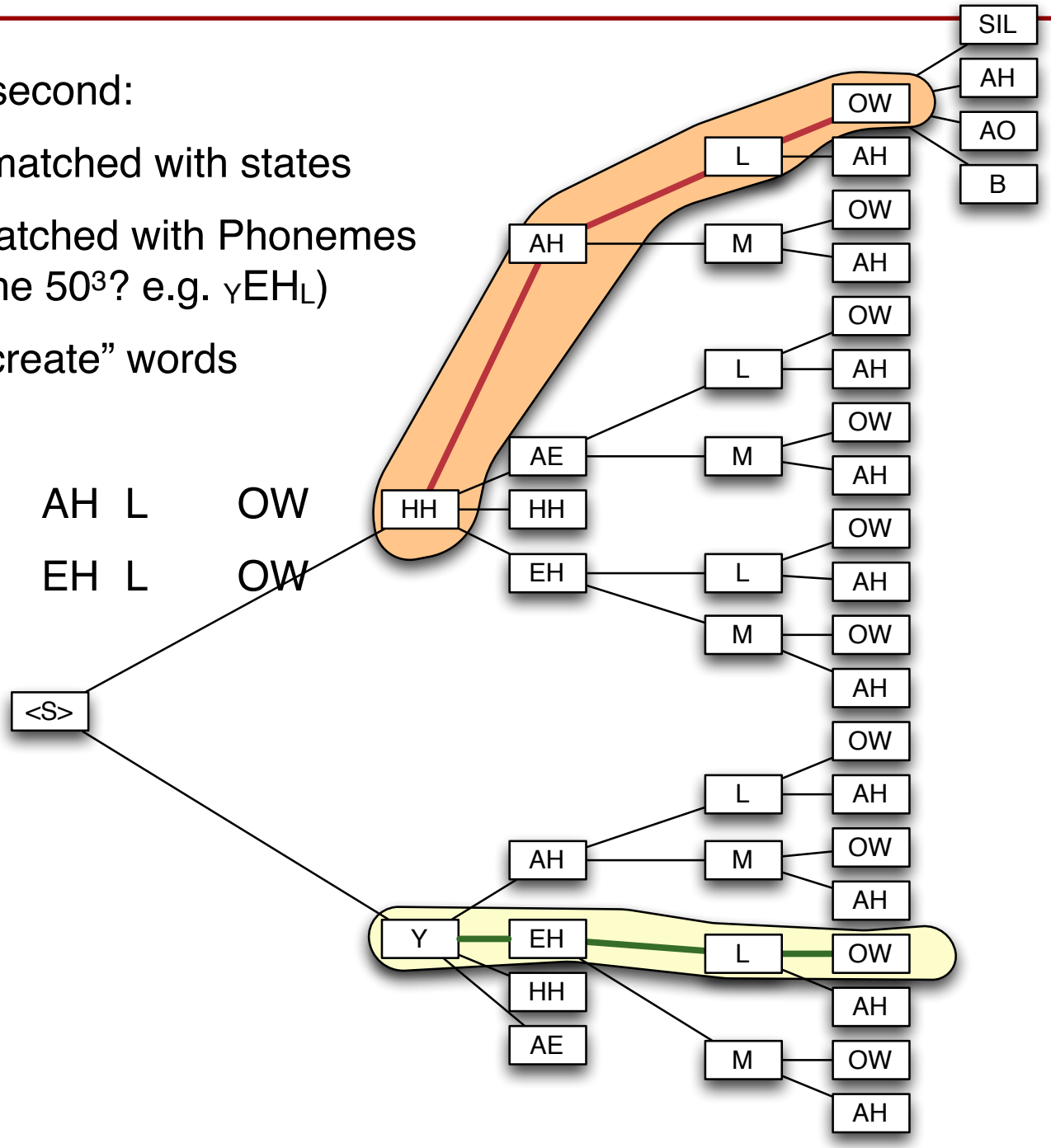
- Other real-system complications:
  - Overlaps
  - Non Verbal Vocalizations
  - Partials
  - Low SNR
  - Denoising
  - Reverberation
  - VAD
  - Diarization
  - Utterance segmentation

Decoding

- 100 times a second:
- Frame gets matched with states
- States get matched with Phonemes (remember the 50^3? e.g. $_{Y}EH_{L}$)
- Phonemes “create” words

HELLO HH AH L OW

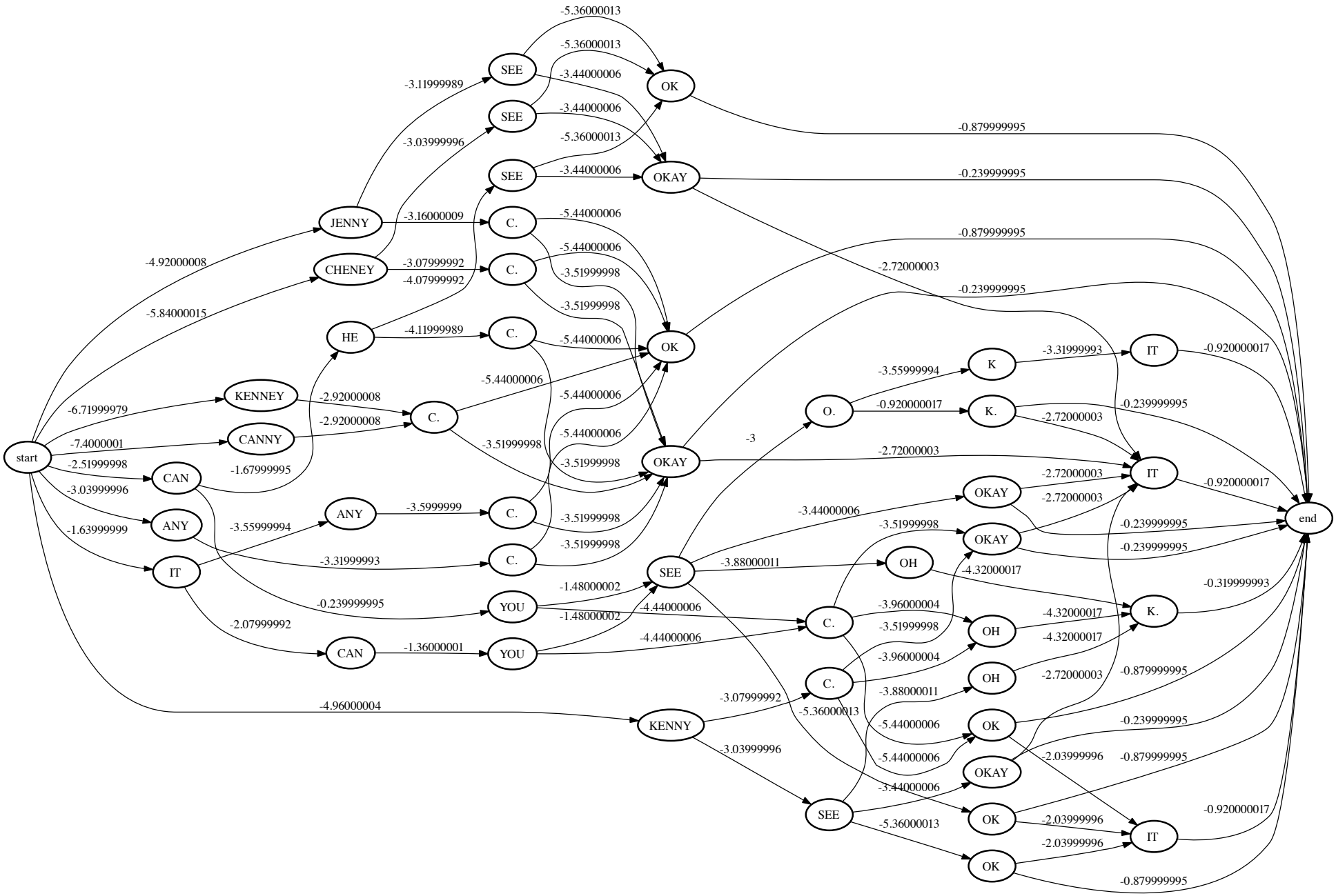
YELLOW Y EH L OW



USC Viterbi School of Engineering

Statistical Signal Processing = uncertainty
=additional information

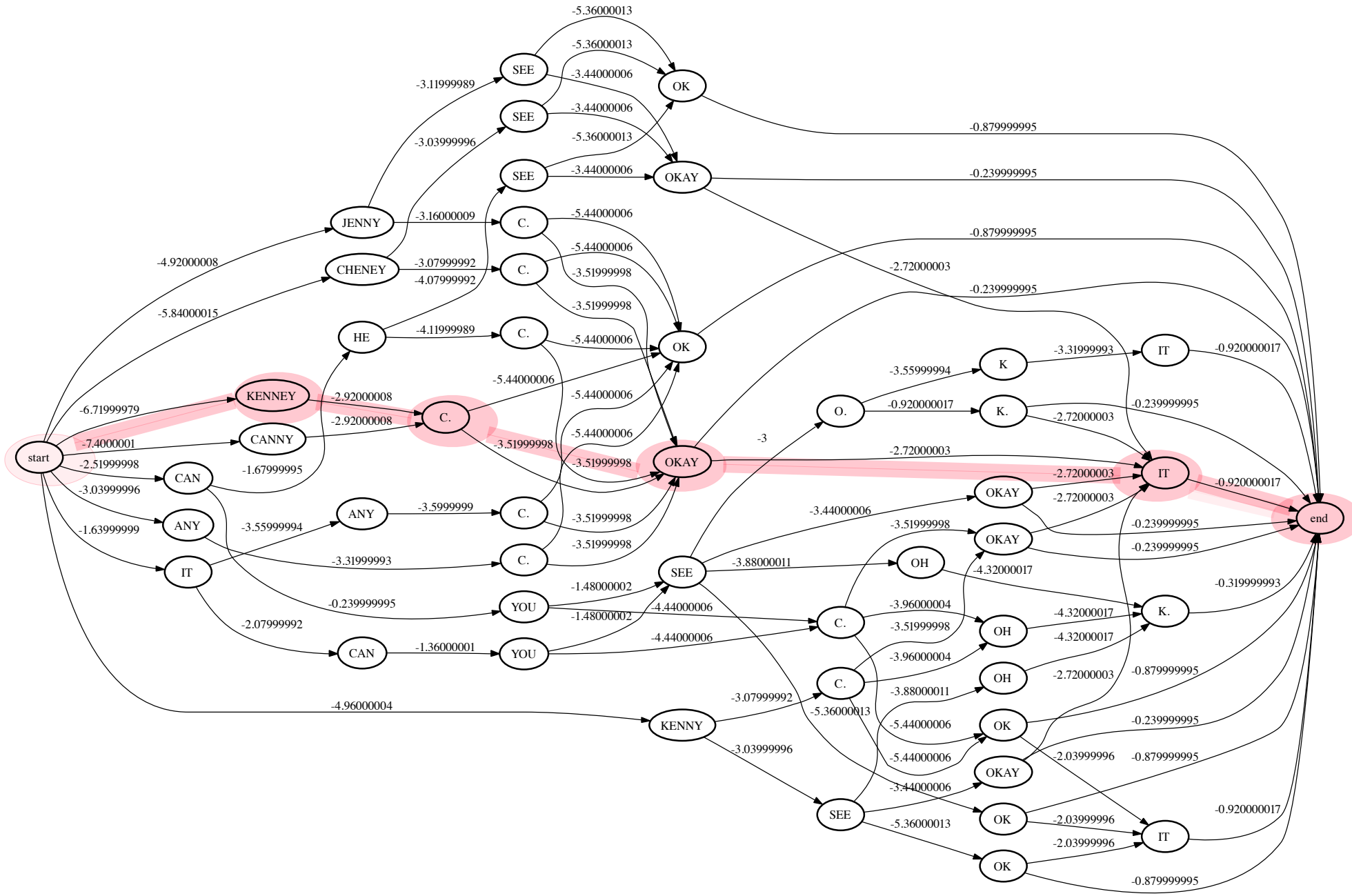
SCUBA



USC Viterbi School of Engineering

Statistical Signal Processing = uncertainty
=additional information

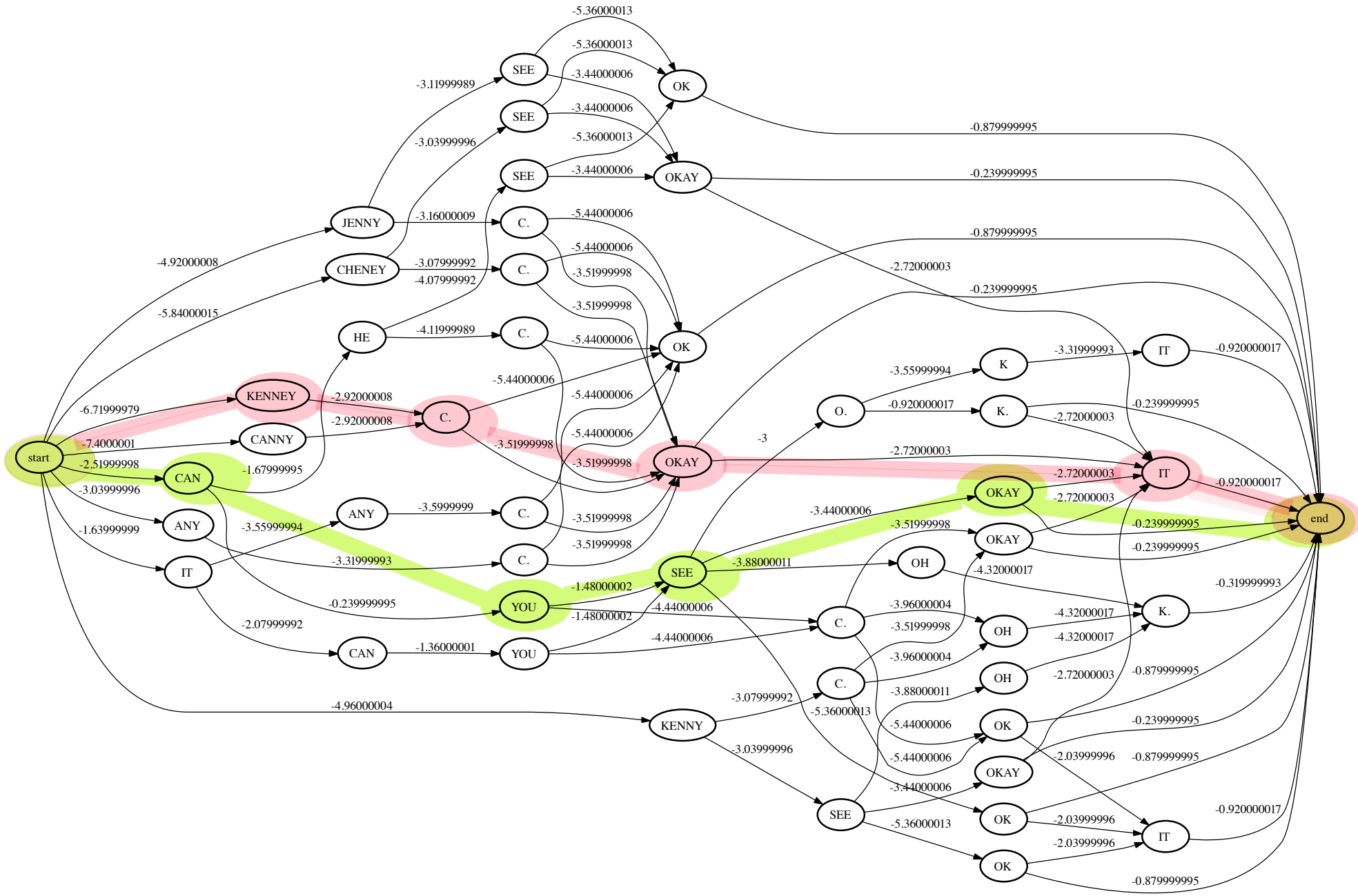
SCUBA



USC Viterbi School of Engineering

Statistical Signal Processing = uncertainty =additional information

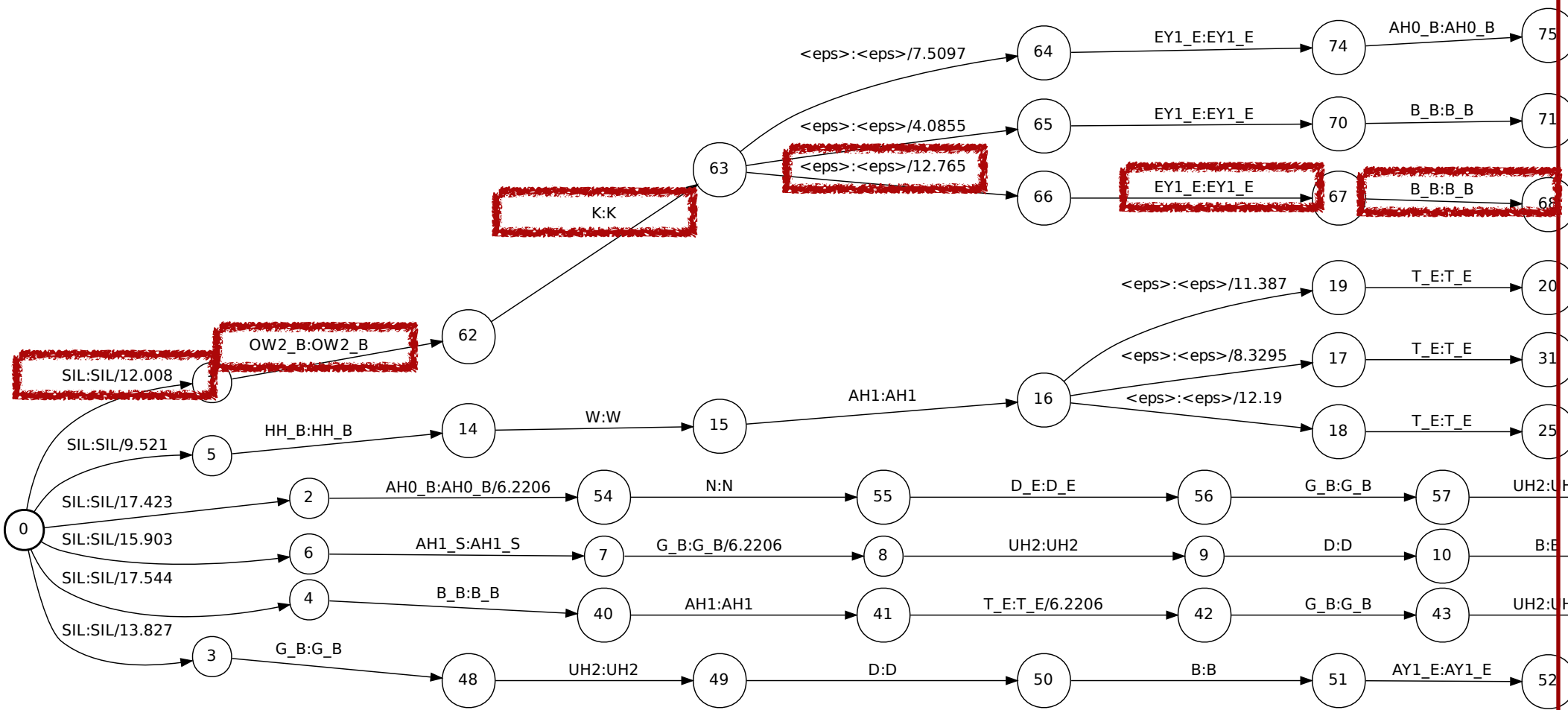
SCUBA



USC Viterbi School of Engineering

The simplest phone lattice

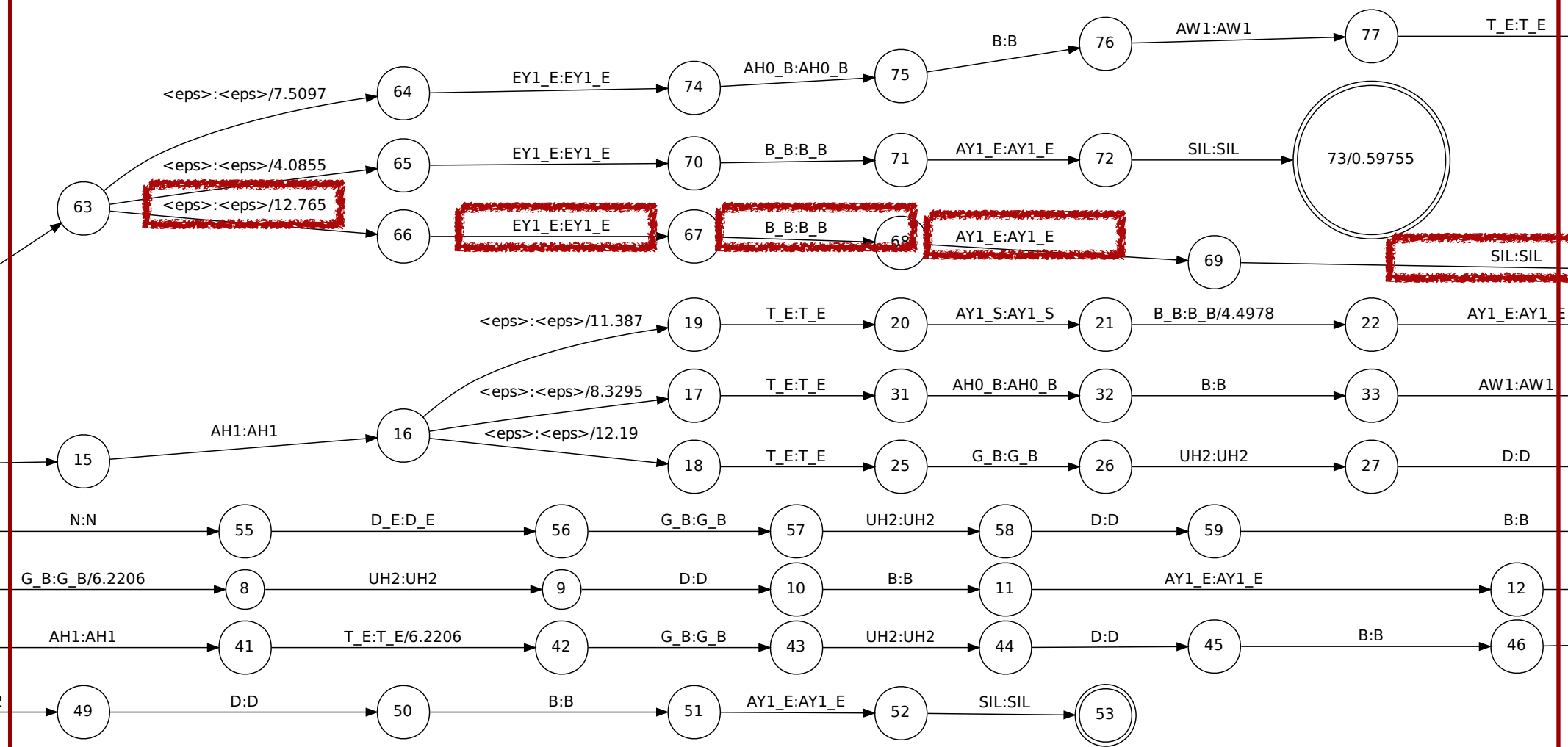
SCUBA



USC Viterbi School of Engineering

The simplest phone lattice

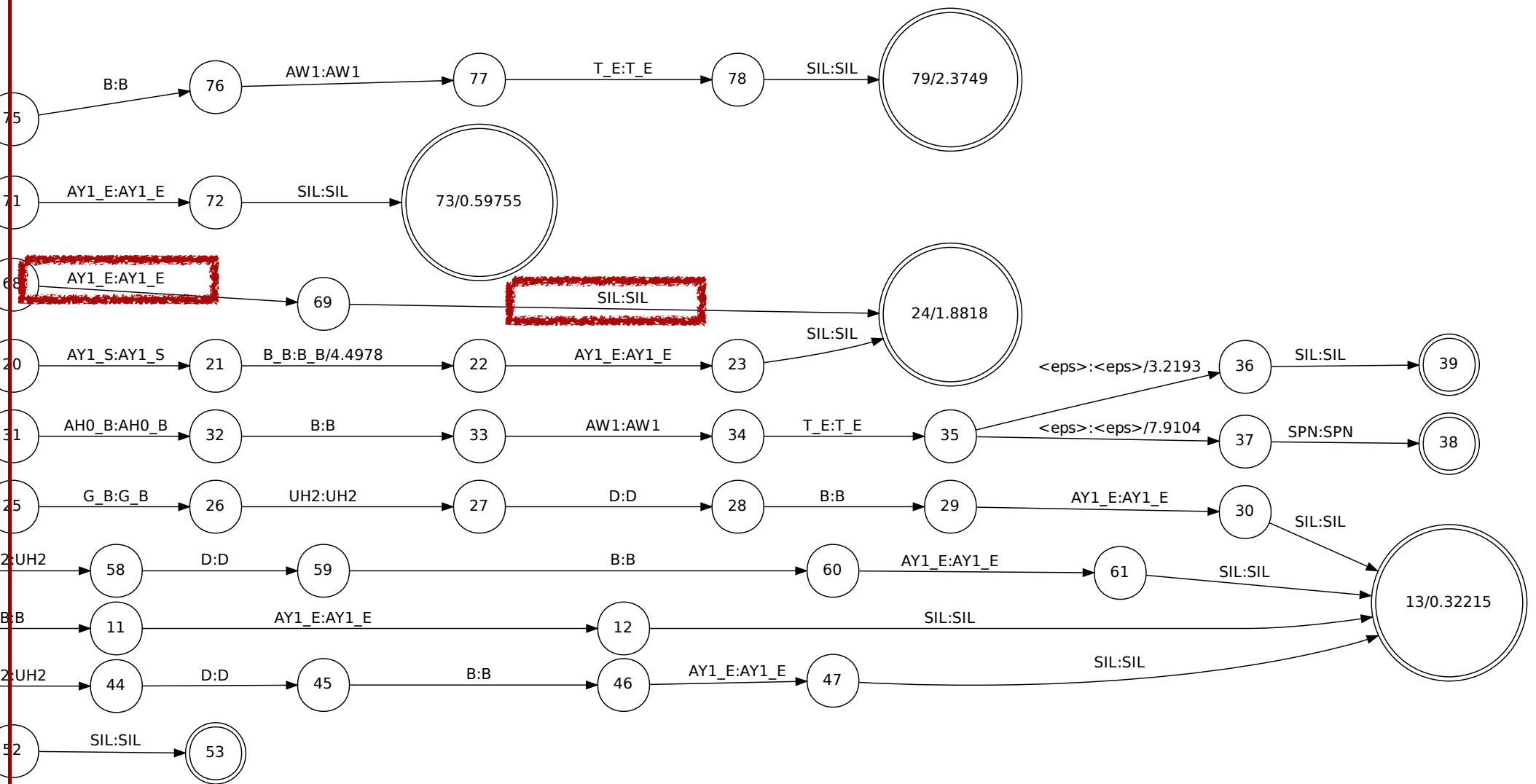
SCUBA

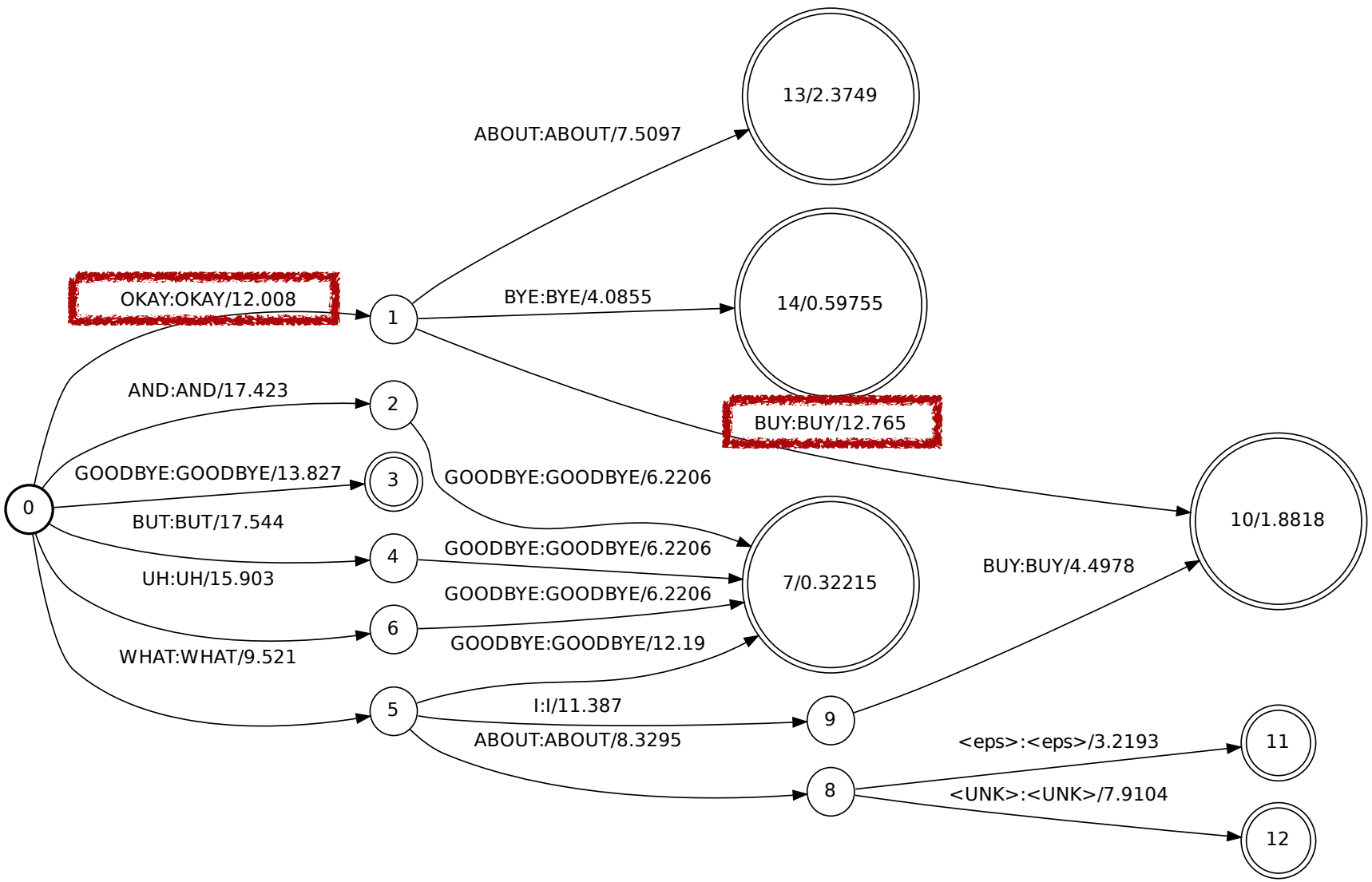


USC Viterbi School of Engineering

The simplest phone lattice

SCUBA

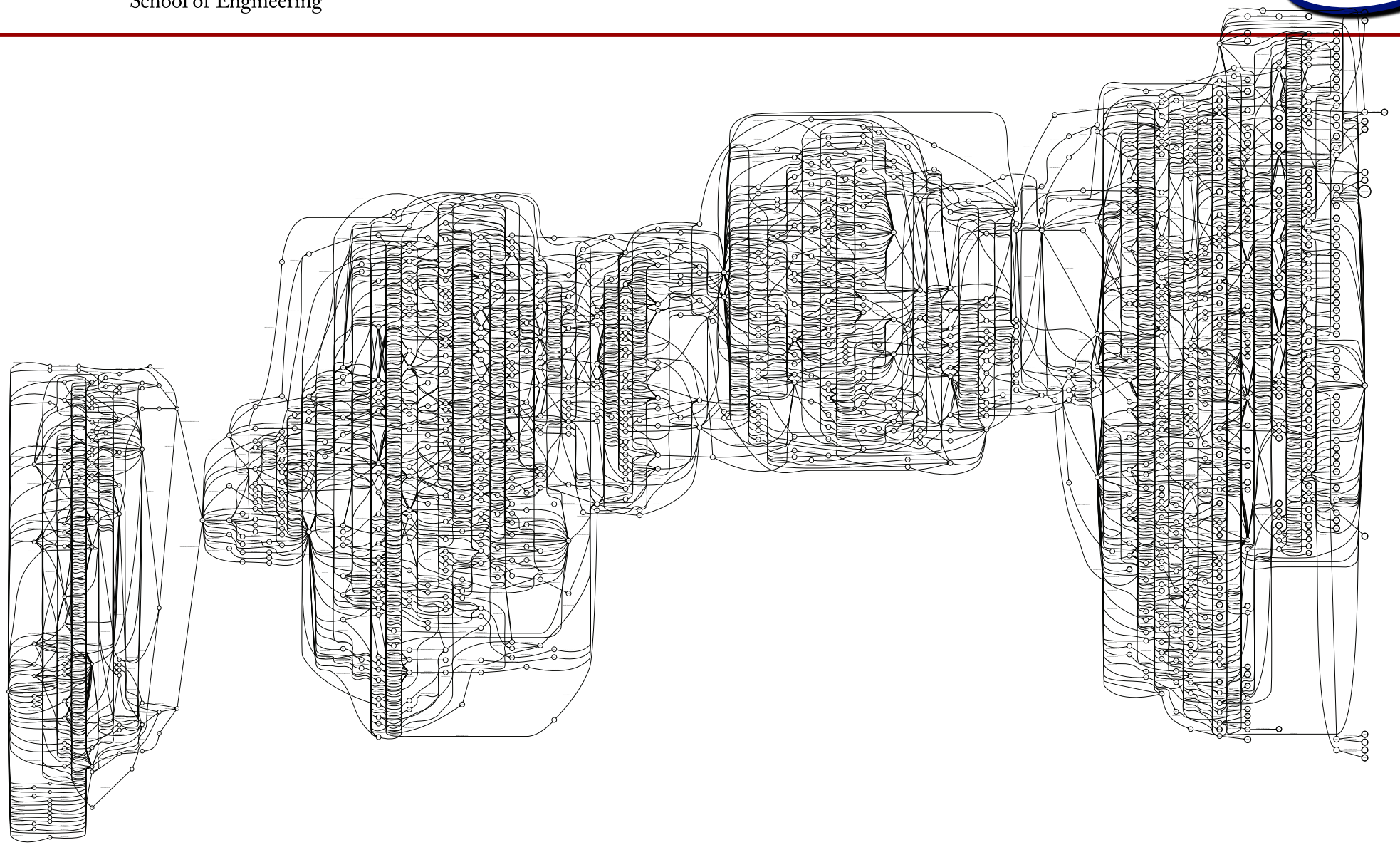



13/2.3749
ABOUT:ABOUT/7.5097
OKAY:OKAY/12.008
1
BYE:BYE/4.0855
14/0.59755
AND:AND/17.423
2
BUY:BUY/12.765
GOODBYE:GOODBYE/13.827
3
GOODBYE:GOODBYE/6.2206
10/1.8818
0
BUT:BUT/17.544
4
GOODBYE:GOODBYE/6.2206
7/0.32215
BUY:BUY/4.4978
UH:UH/15.903
GOODBYE:GOODBYE/6.2206
6
GOODBYE:GOODBYE/12.19
WHAT:WHAT/9.521
5
I:I/11.387
9
ABOUT:ABOUT/8.3295
<eps>:<eps>/3.2193
11
8
<UNK>:<UNK>/7.9104
12

USC Viterbi School of Engineering

Real Real Lattice

SCUBA

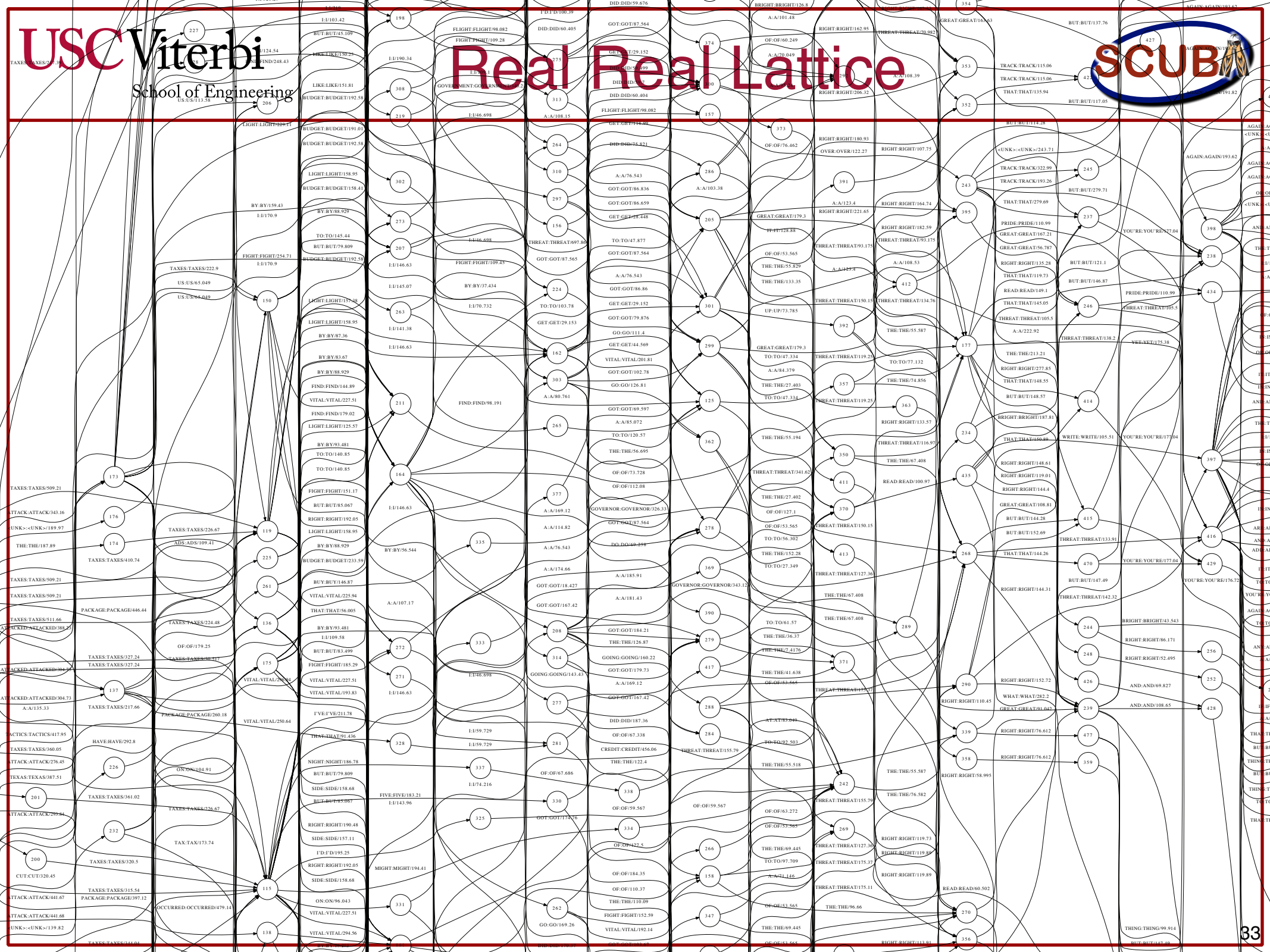


Real Real Lattice

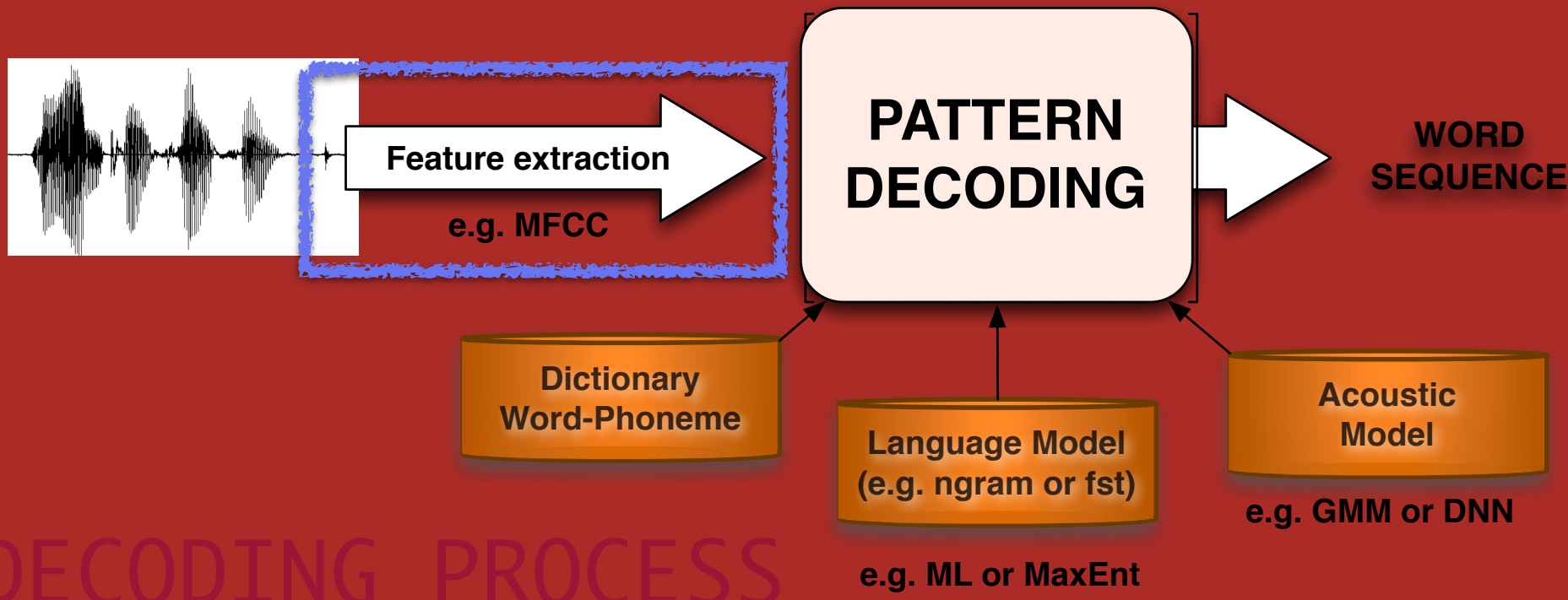
USC Viterbi School of Engineering

Real Real Lattice

SCUBA



Speech Production 101



DECODING PROCESS

USC Viterbi School of Engineering

Speech Production

SCUBA

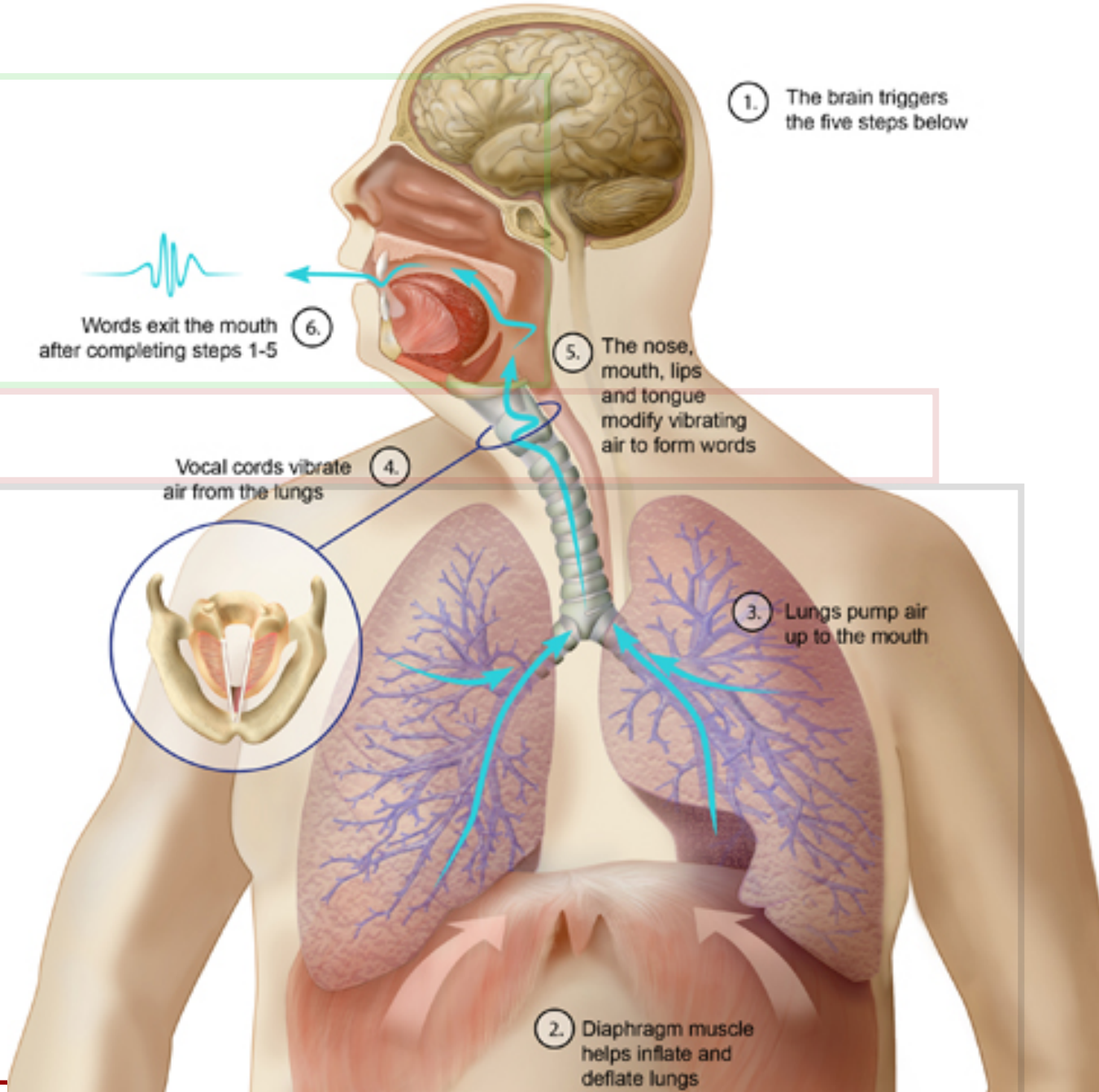
Supra-Laryngeal
Vocal Tract

Filter

Larynx
Excitation

Sub-glottal
System

Source



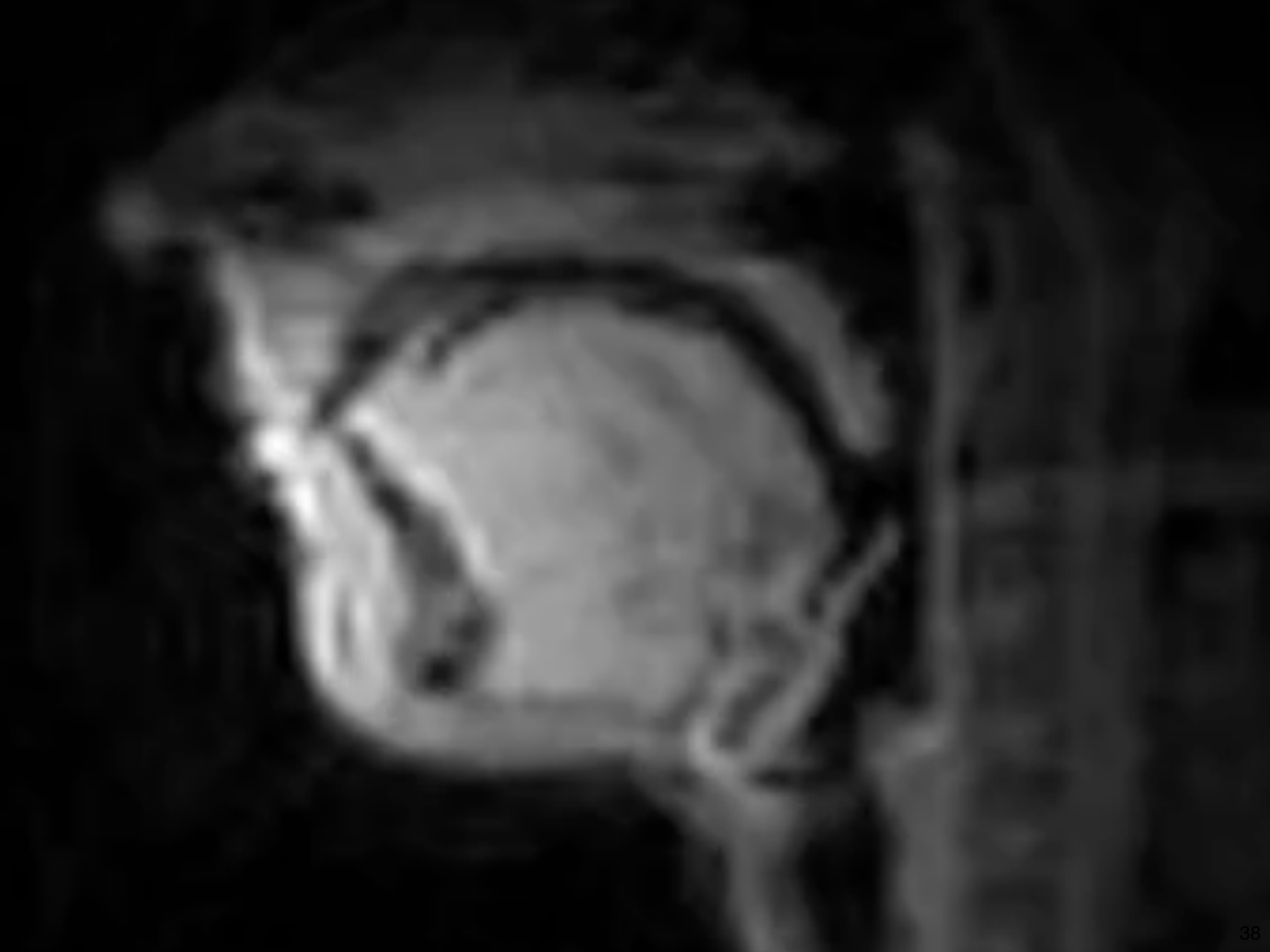
The Source-Filter Model of Speech Production (Chiba & Kajiyama, 1940)

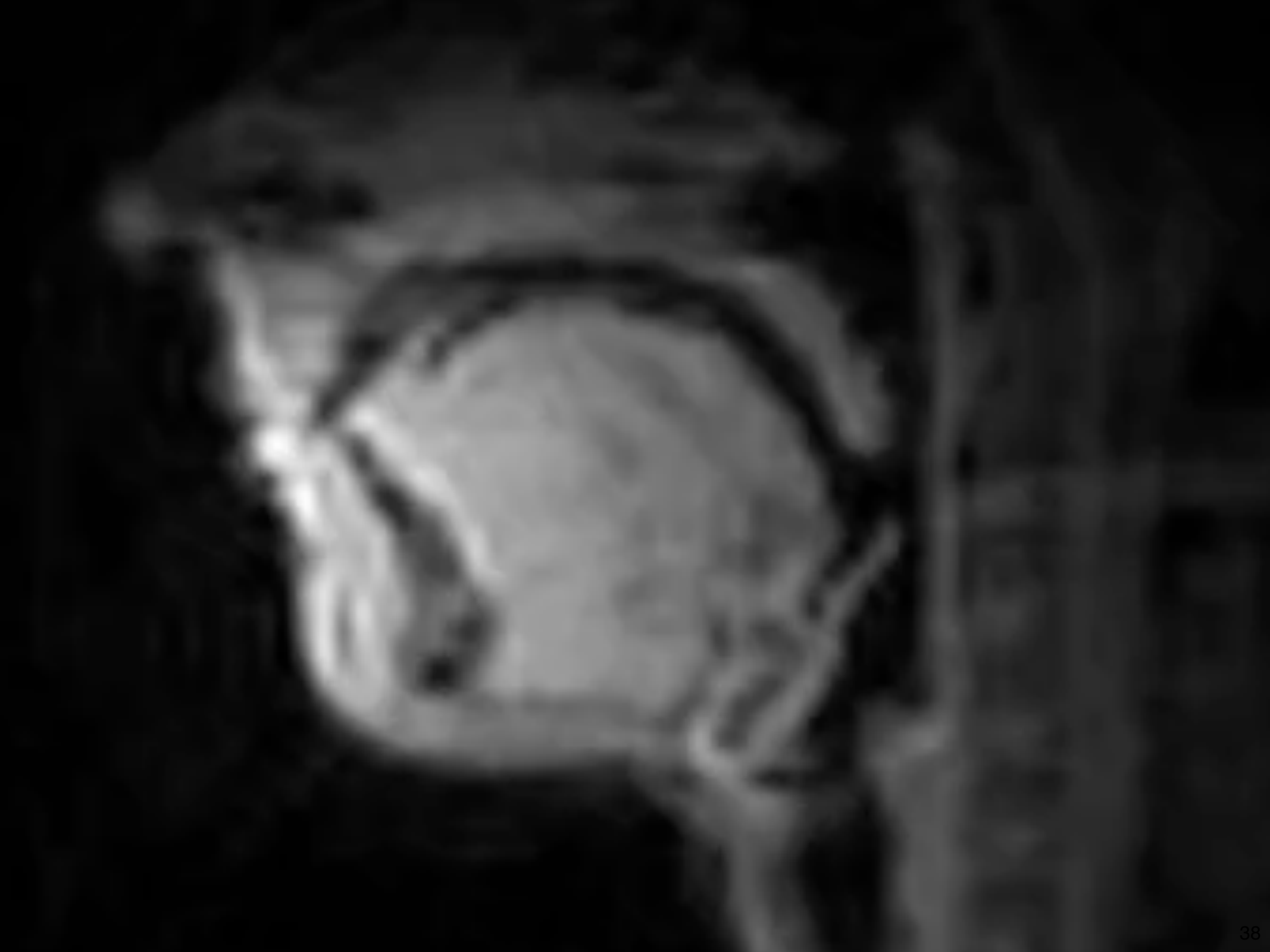
- Sources: 3 & all of them have wideband spectrum
  - Voicing: vibration of the vocal folds, same type of aerodynamic mechanism as a flag flapping in the wind.
  - Frication or Aspiration: turbulence created when air passes through a narrow aperture
  - Burst: the “pop” that occurs when high air pressure is suddenly released

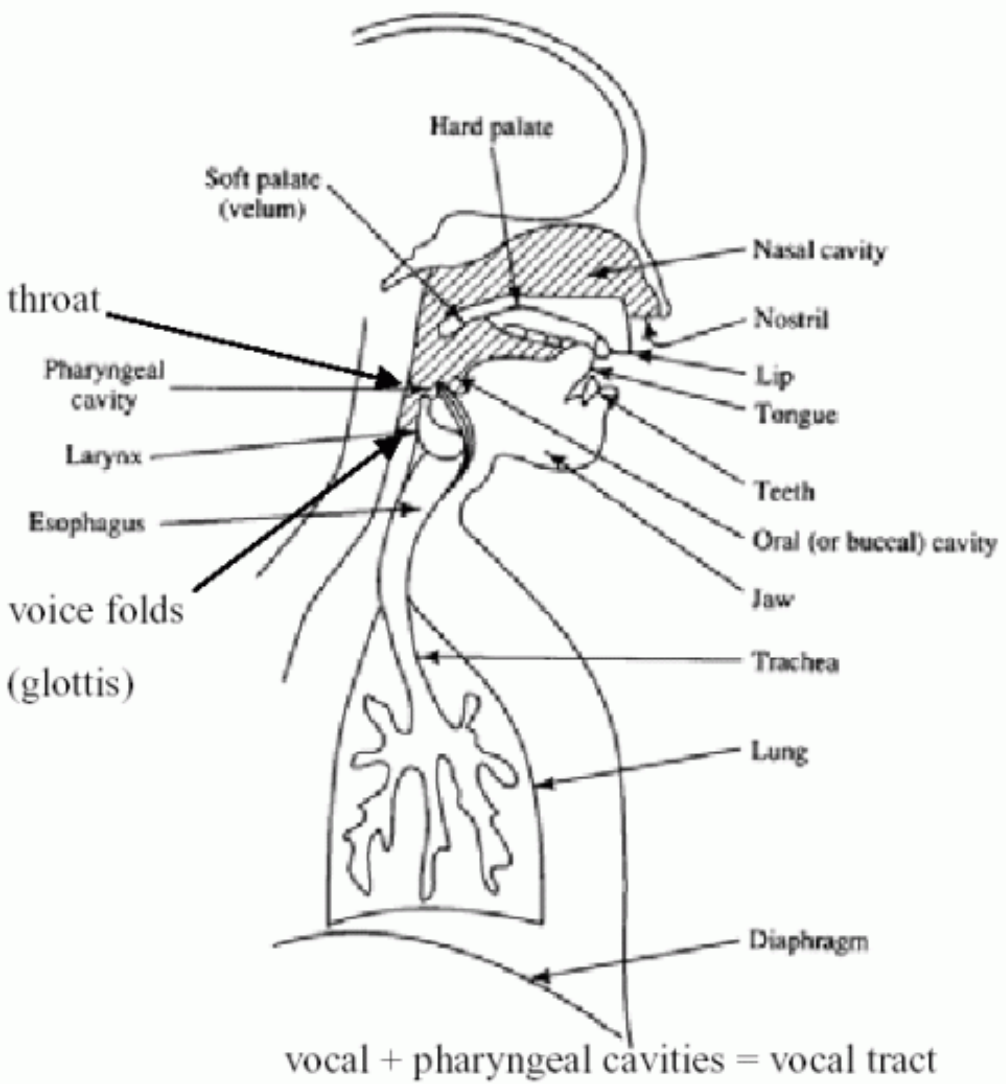
- Filter:
  - Vocal tract = the air cavity between glottis and lips
  - Just like a flute or a shower stall, it has resonances
  - The excitation has energy at all frequencies; excitation at the resonant frequencies is enhanced

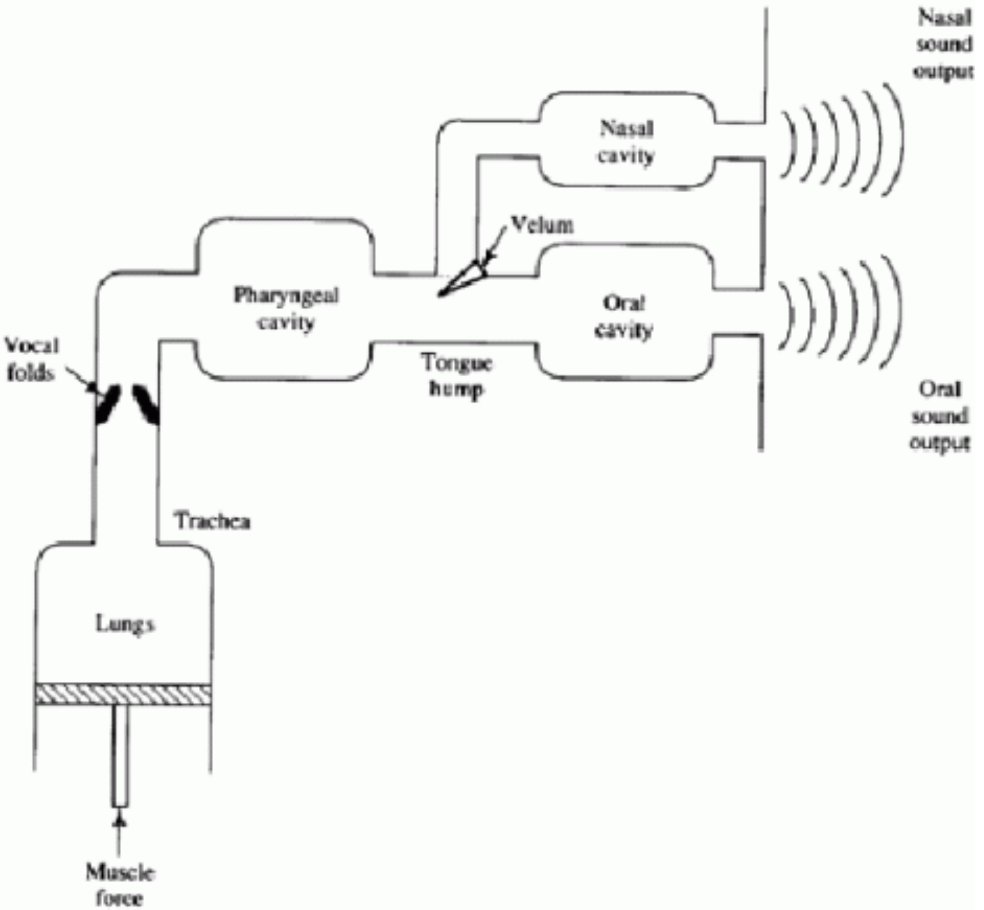
Why 3 states in Phoneme

- To produce speech it takes 3 actions:
  - Initiation
  - Phonation
  - Articulation: The distortion through a ‘tube’
- (and clearly a very important previous action in the brain!)





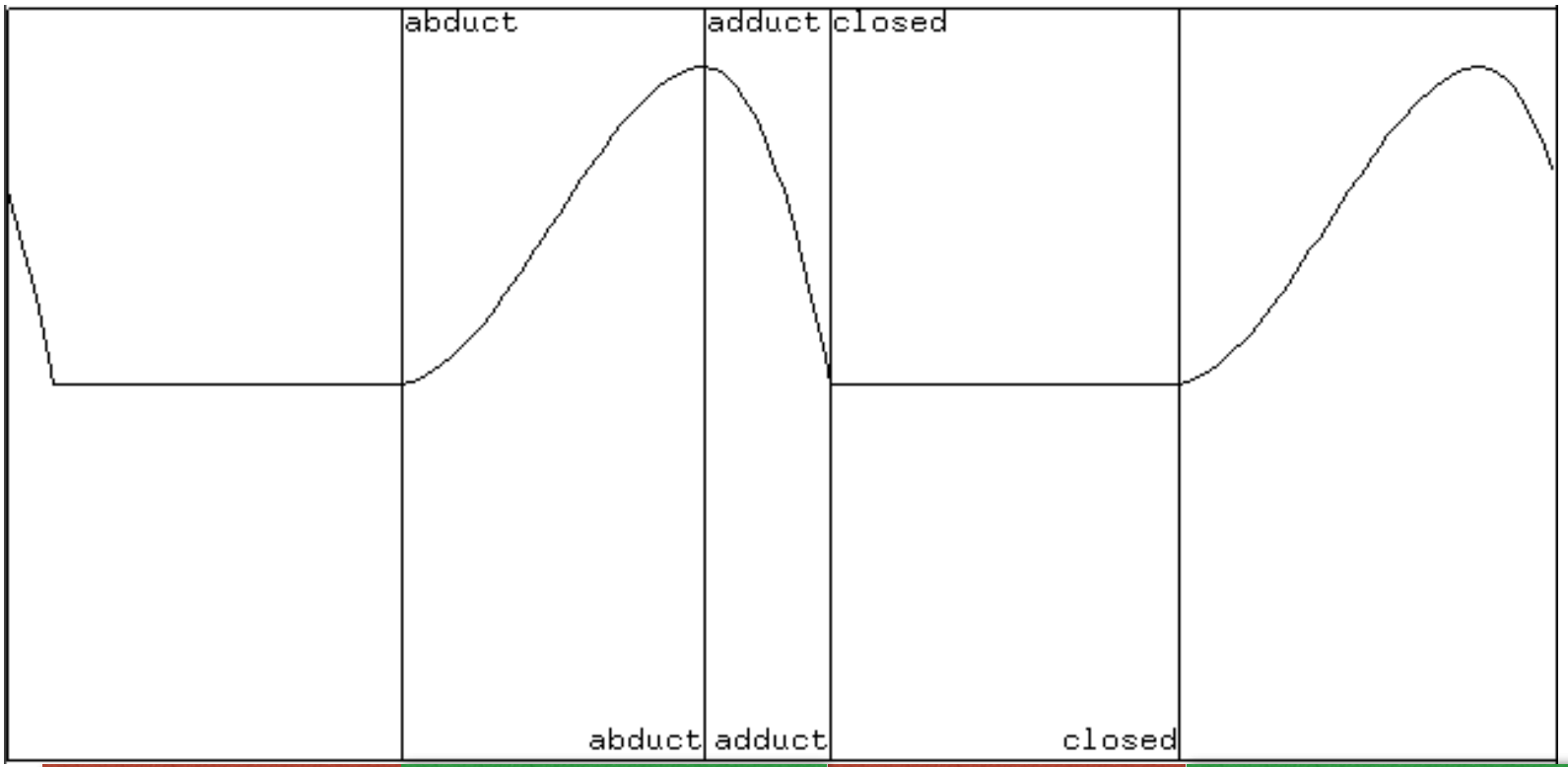

Hard palate
Soft palate (velum)
Nasal cavity
throat
Nostril
Pharyngeal cavity
Lip
Tongue
Larynx
Teeth
Esophagus
Oral (or buccal) cavity
voice folds
Jaw
(glottis)
Trachea
Lung
Diaphragm
vocal + pharyngeal cavities = vocal tract


Nasal sound output
Nasal cavity
Velum
Pharyngeal cavity
Oral cavity
Vocal folds
Tongue hump
Oral sound output
Trachea
Lungs
Muscle force

- EE 519 looks at the speech production system in detail
- In short:
  - The source-excitation-filter model is a rough approximation
  - There are many shortcomings:
  - 3D wave propagation
  - Temporal changes of vocal tract shape (emotions!)
  - viscous fiction at the walls
  - softness of the tract walls
  - radiation of sound at the lips
  - nasal coupling
  - excitation of sound

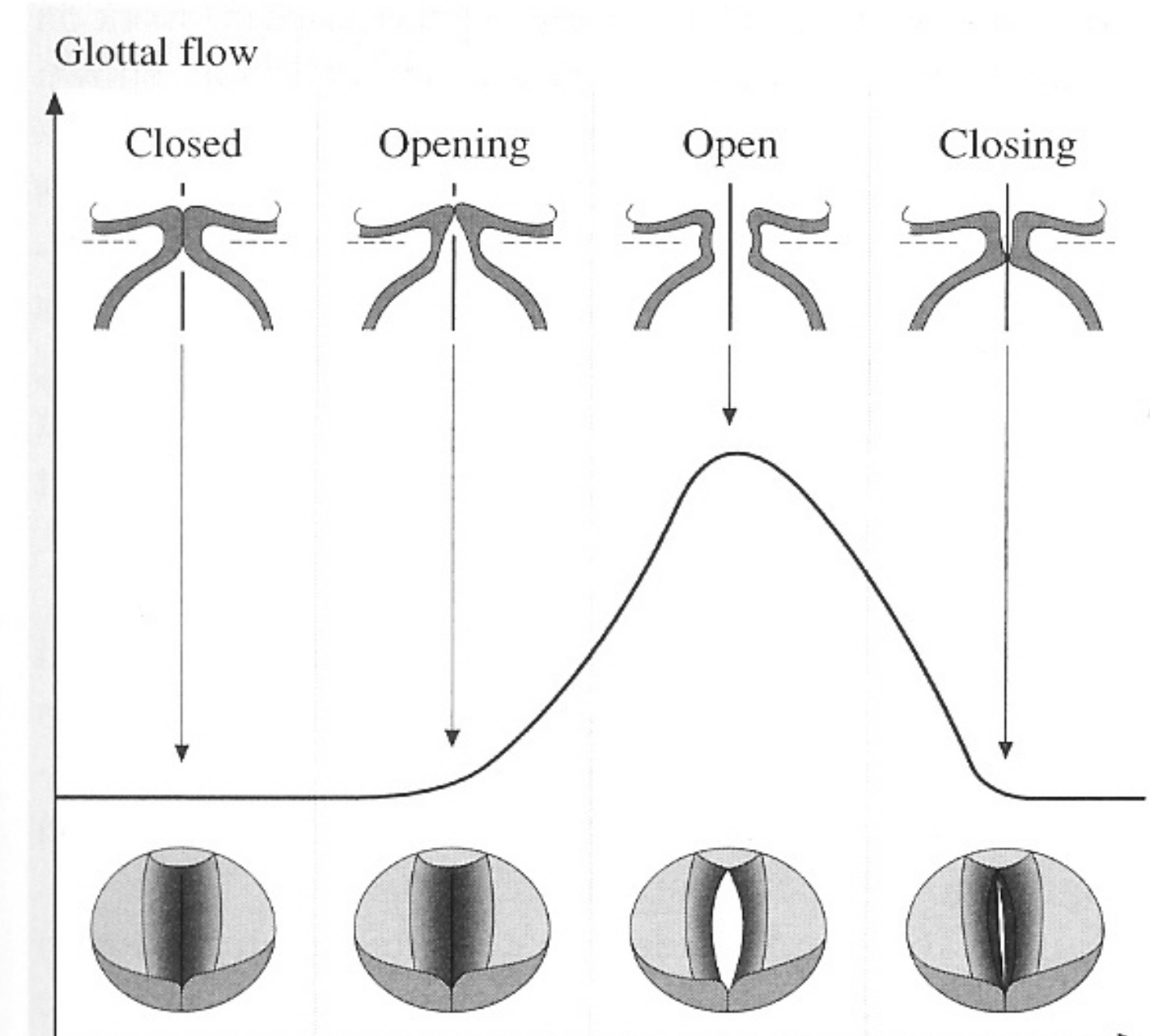
Glottal flow

- Air volume vs time (Rosenberg's model)
- The frequency of this is the Fundamental Frequency (f_0)



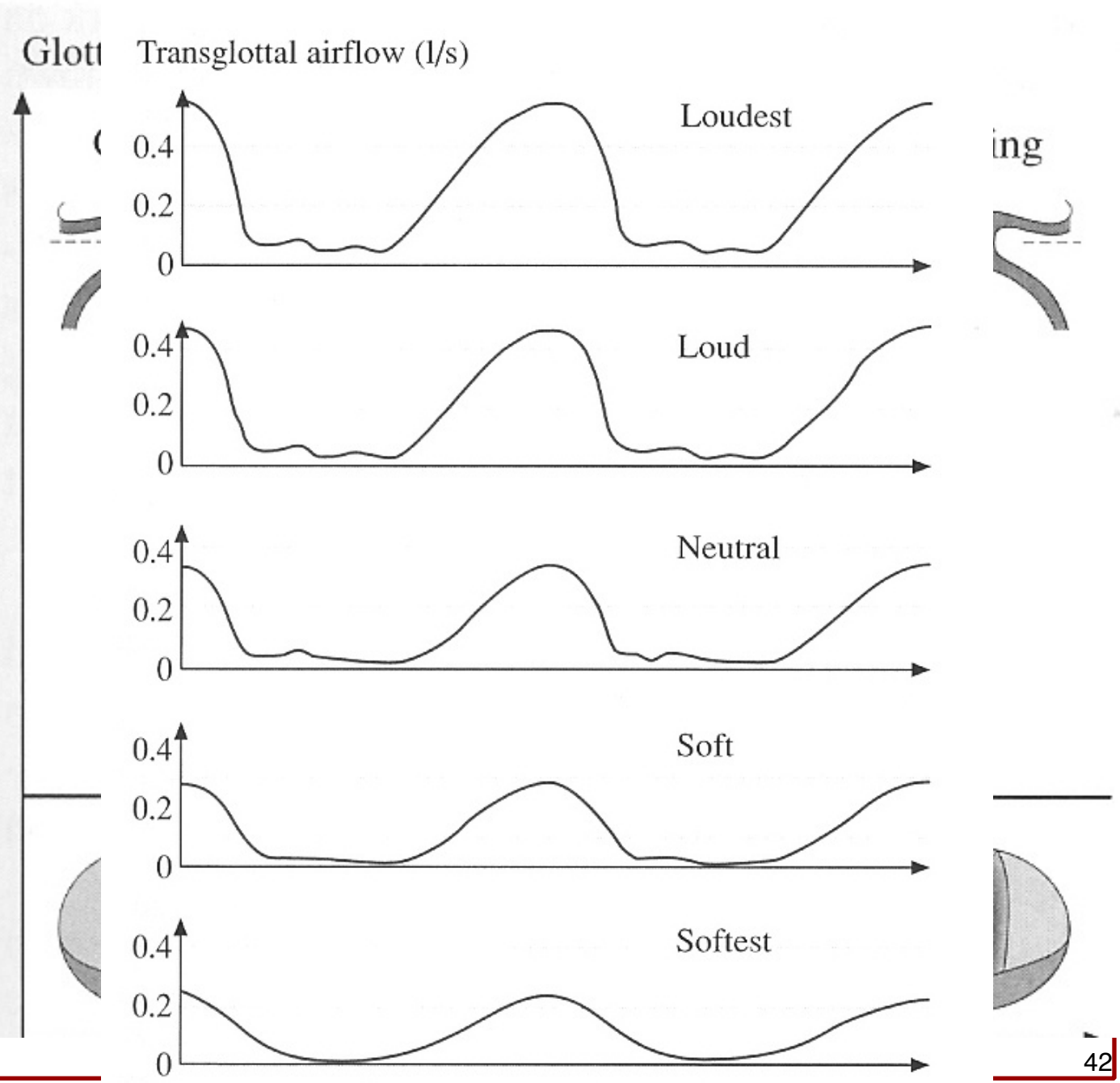
Glottogram

- Not really a nice, predictable shape
- Relating to behavioral aspects, conditions, e.g. emotions, nervousness, parkinson's, …



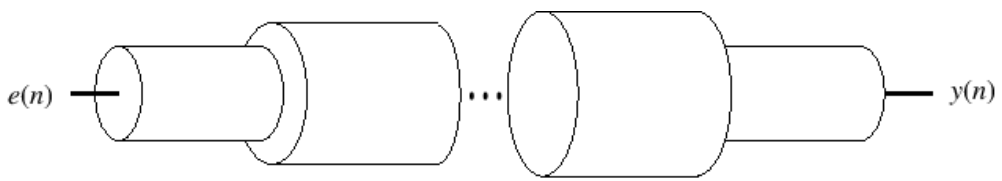
Glottogram

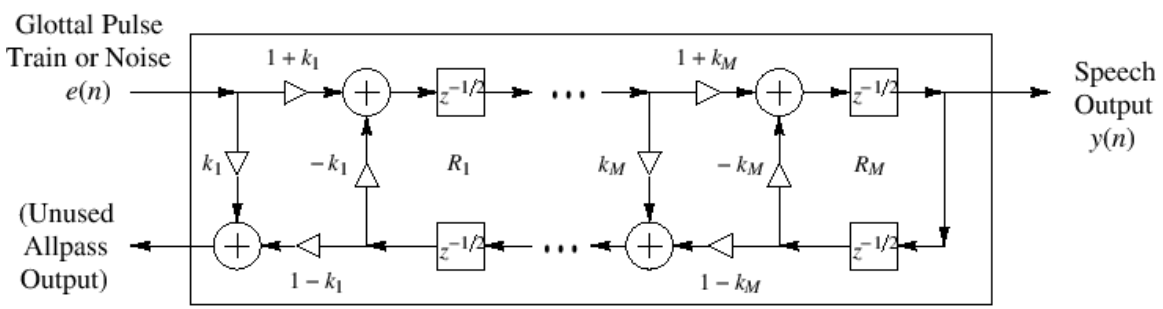
- Not really a nice, predictable shape
- Relating to behavioral aspects, conditions, e.g. emotions, nervousness, parkinson's, …



Lossless Tube Concatenation Model

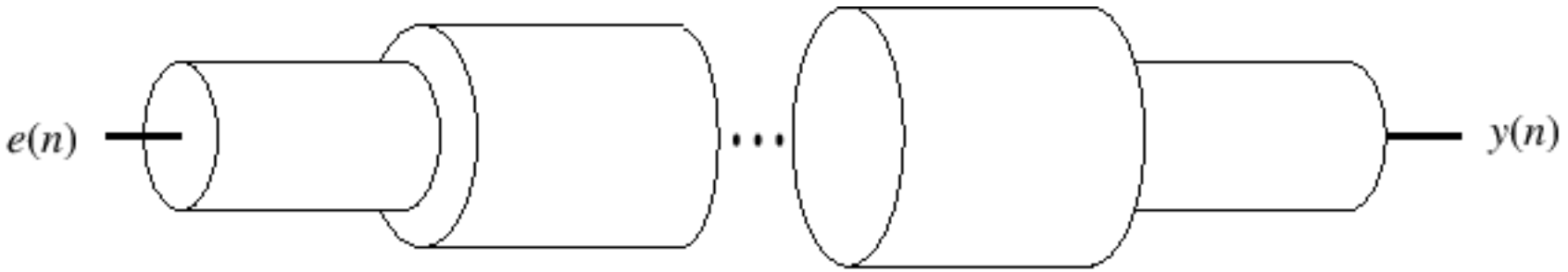
- Idea: The vocal tract can be represented as a concatenation of lossless tubes
- It consists of a series of cylinders (Helmholtz-Resonator) of equal length
- The cross-sections approximate the area function of the vocal tract
- If many tubes is large of short length (nyquist), the frequency response is expected to be close to those of tubes with continuously varying area functions
- For waves with wavelength >> dimensions of the vocal tract, waves propagate along the axis of the tubes
- Further assume: No loss due to viscosity or thermal conduction, and area A remains constant over time

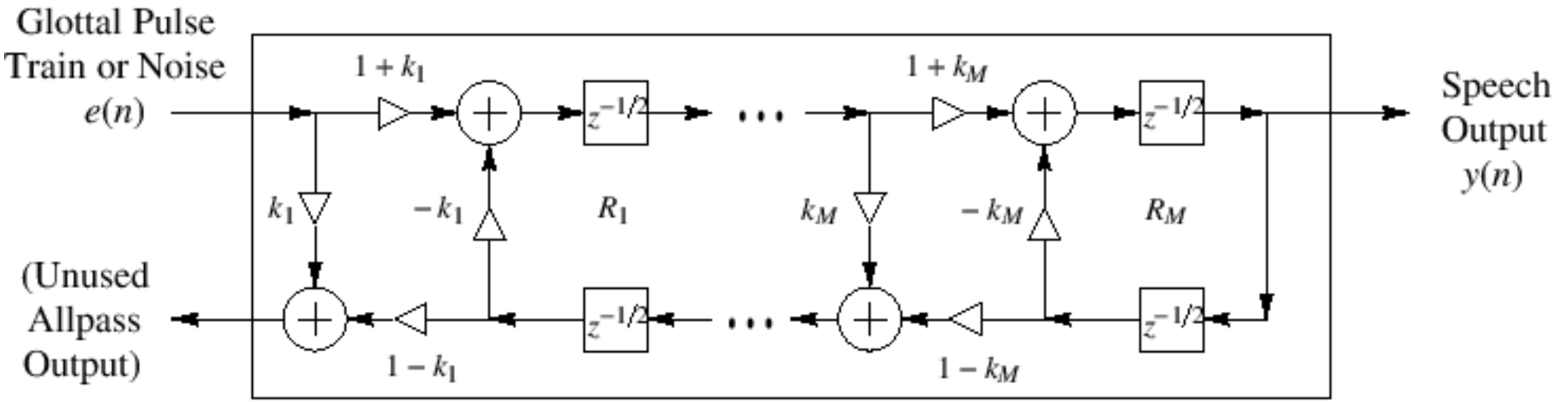




Kelly-Lochbaum Vocal Tract Model (Piecewise Cylindrical)

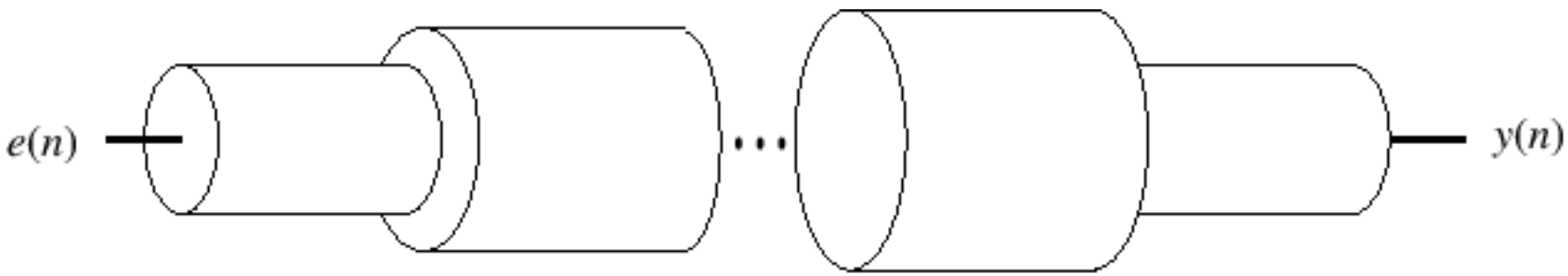
Lossless Tube Concatenation Model



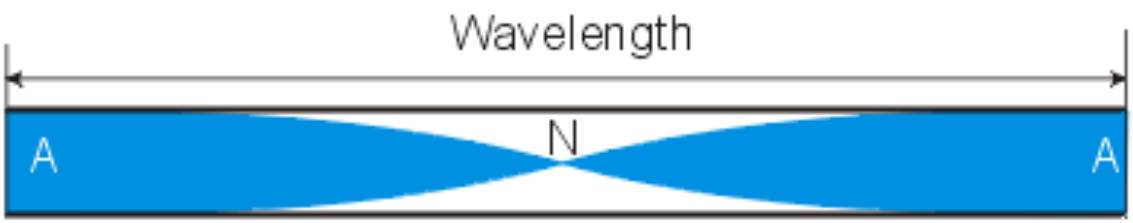


Kelly-Lochbaum Vocal Tract Model (Piecewise Cylindrical)

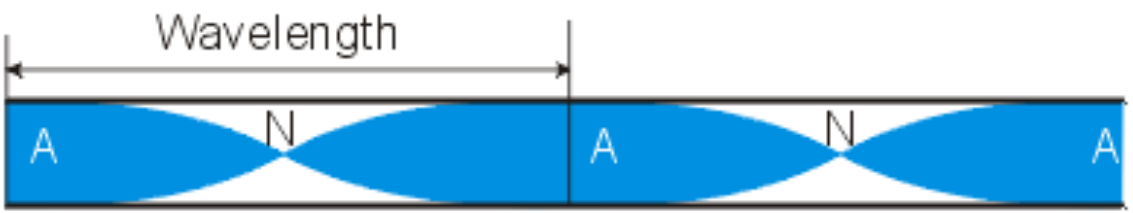
Lossless Tube Concatenation Model



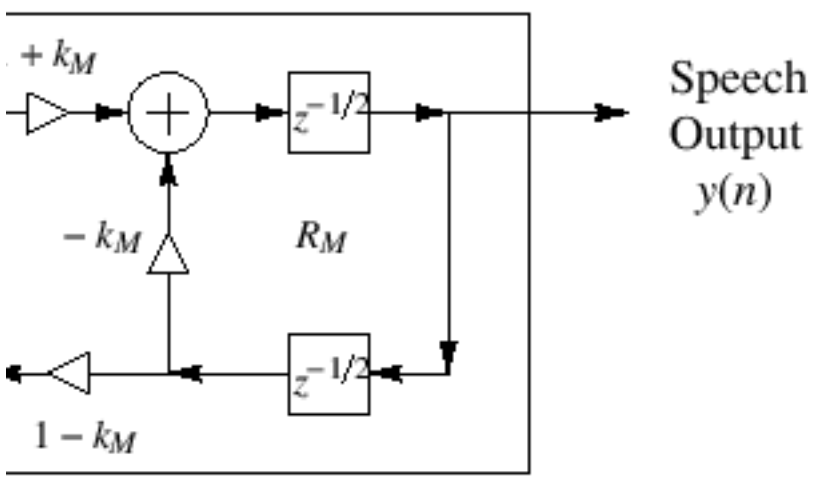
Standing wave in an open pipe



Fundamental wave

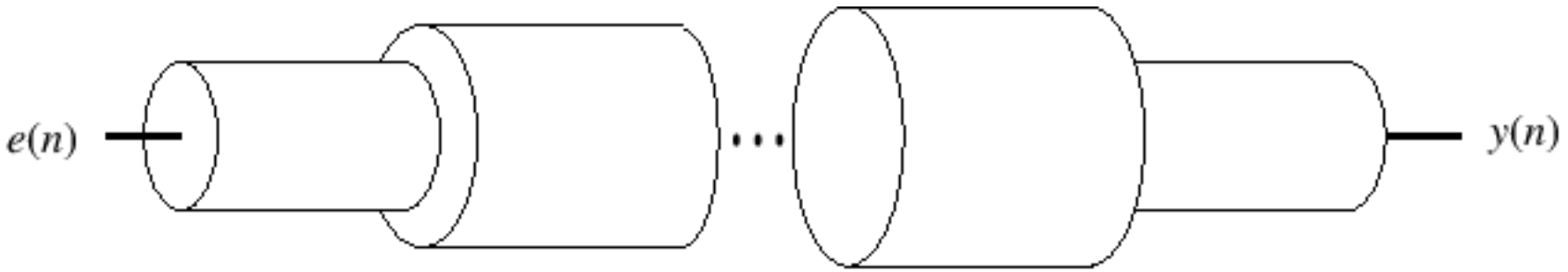


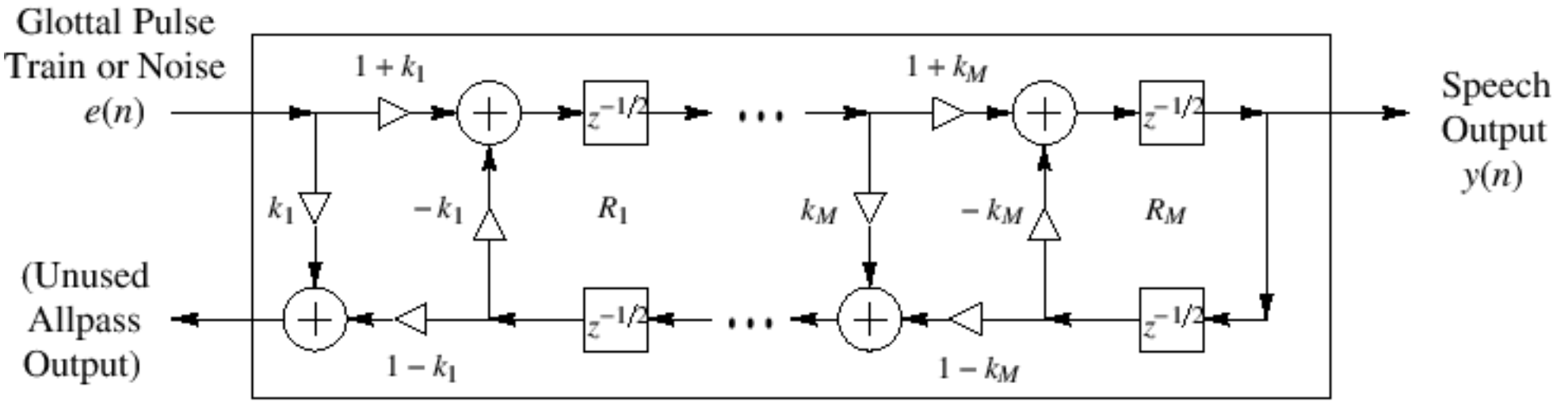
A harmonic wave



Piecewise Cylindrical)

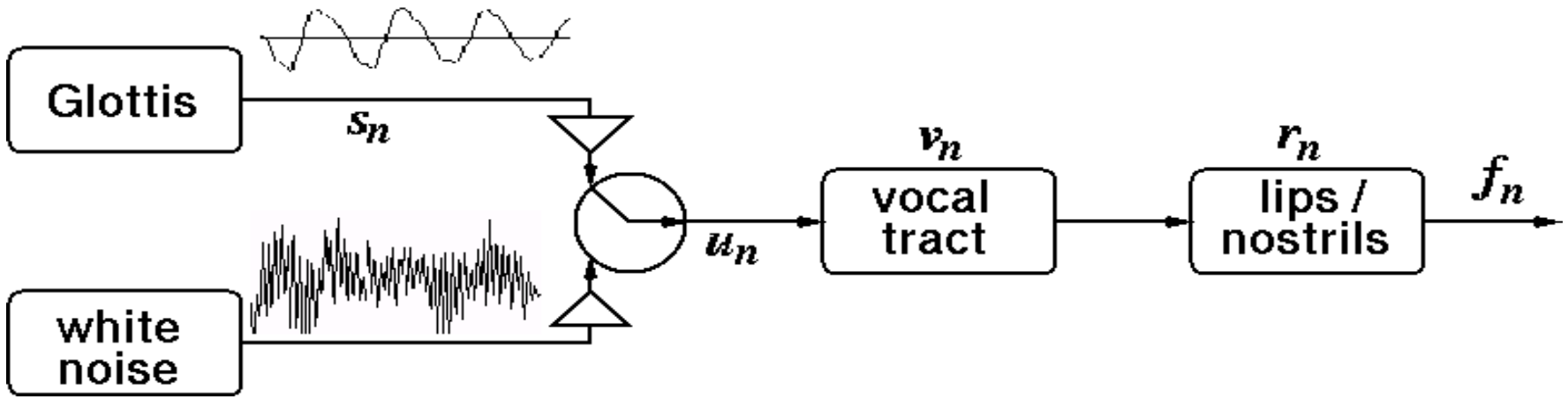
Lossless Tube Concatenation Model





Kelly-Lochbaum Vocal Tract Model (Piecewise Cylindrical)

The Source-Filter Model

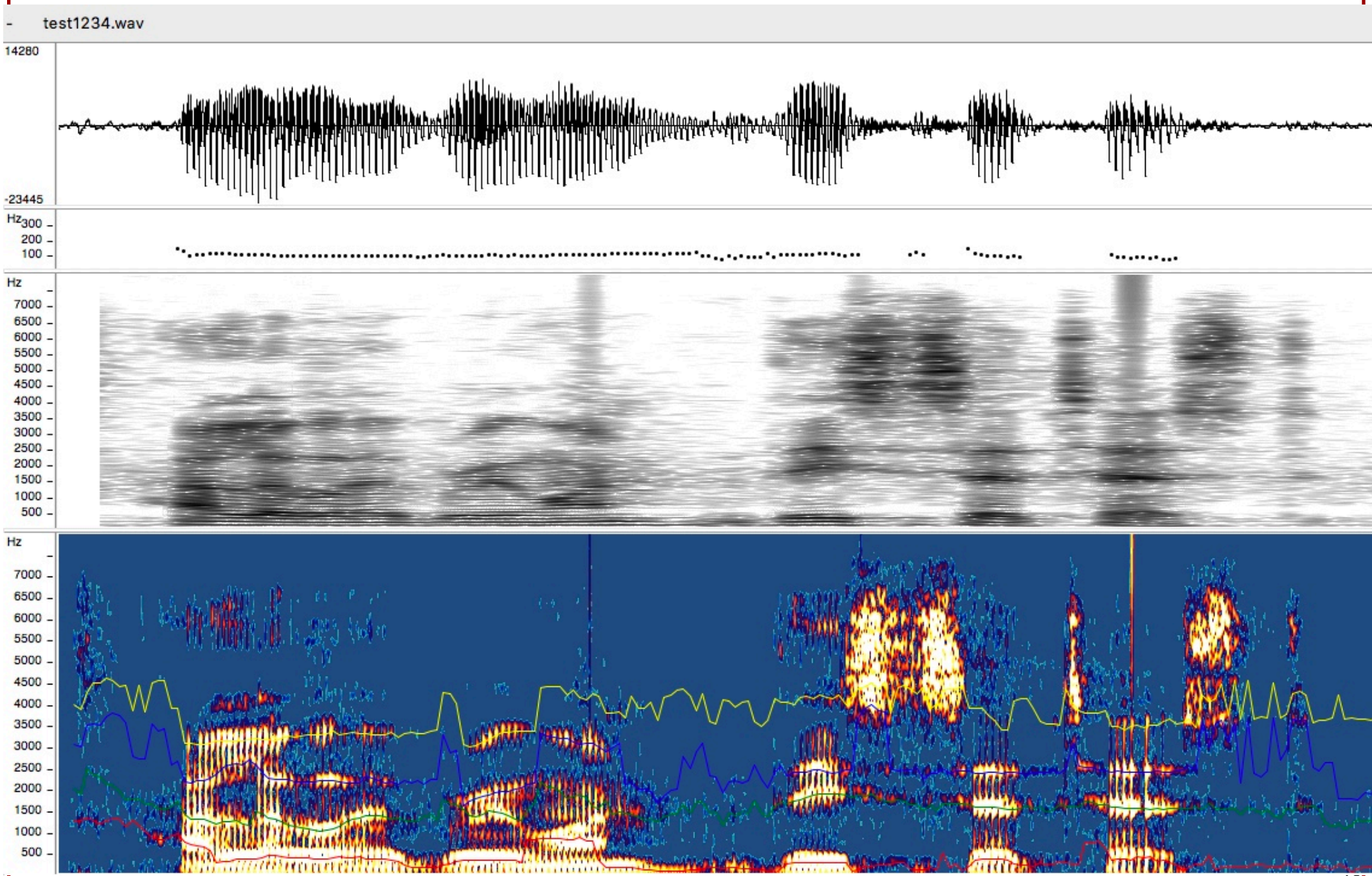


- Sounds are produced by either
  - vibrating the vocal cords (voiced sounds — run audacity demo) or
  - random noise resulting from friction of the airflow (unvoiced sounds)
  - voiced fricatives need a mixed excitation model
- The signal un is modulated by the vocal tract, whose impulse response is vn
- Resulting signal is modulated by the lips' and nostrils' radiation response rn.
- Eventually, the resulting signal fn is emitted.
- The modulation yn of a signal xn by a channel c can be computed as the convolution of the signal with the channel's impulse response $y_n = x_n * c_n$
  Thus: $f_n = u_n * v_n * r_n$

USC Viterbi School of Engineering

Formants, Pitch, Waveform, Spectrum

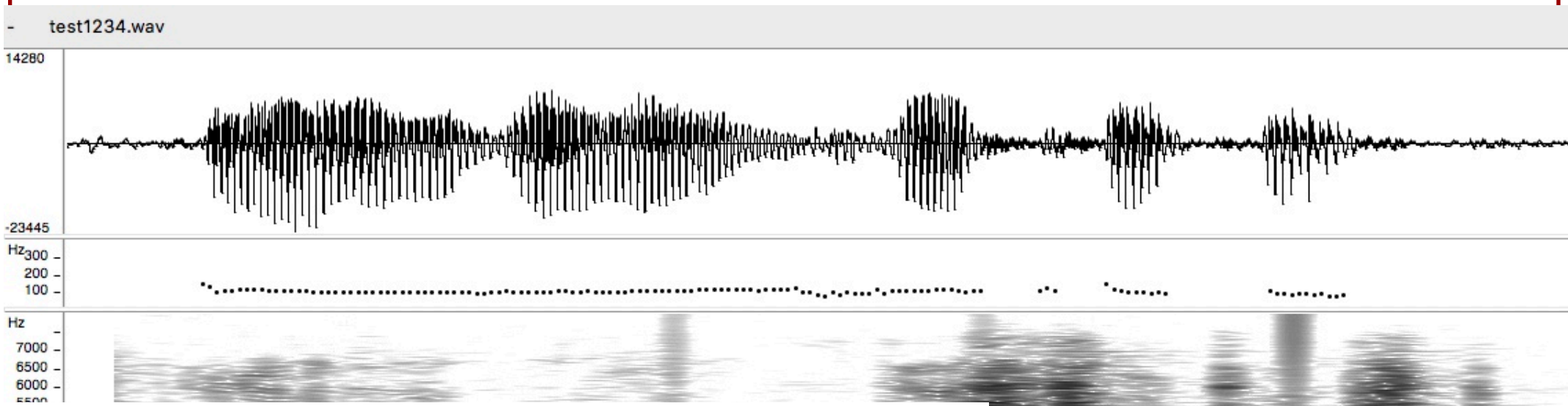
SCUBA

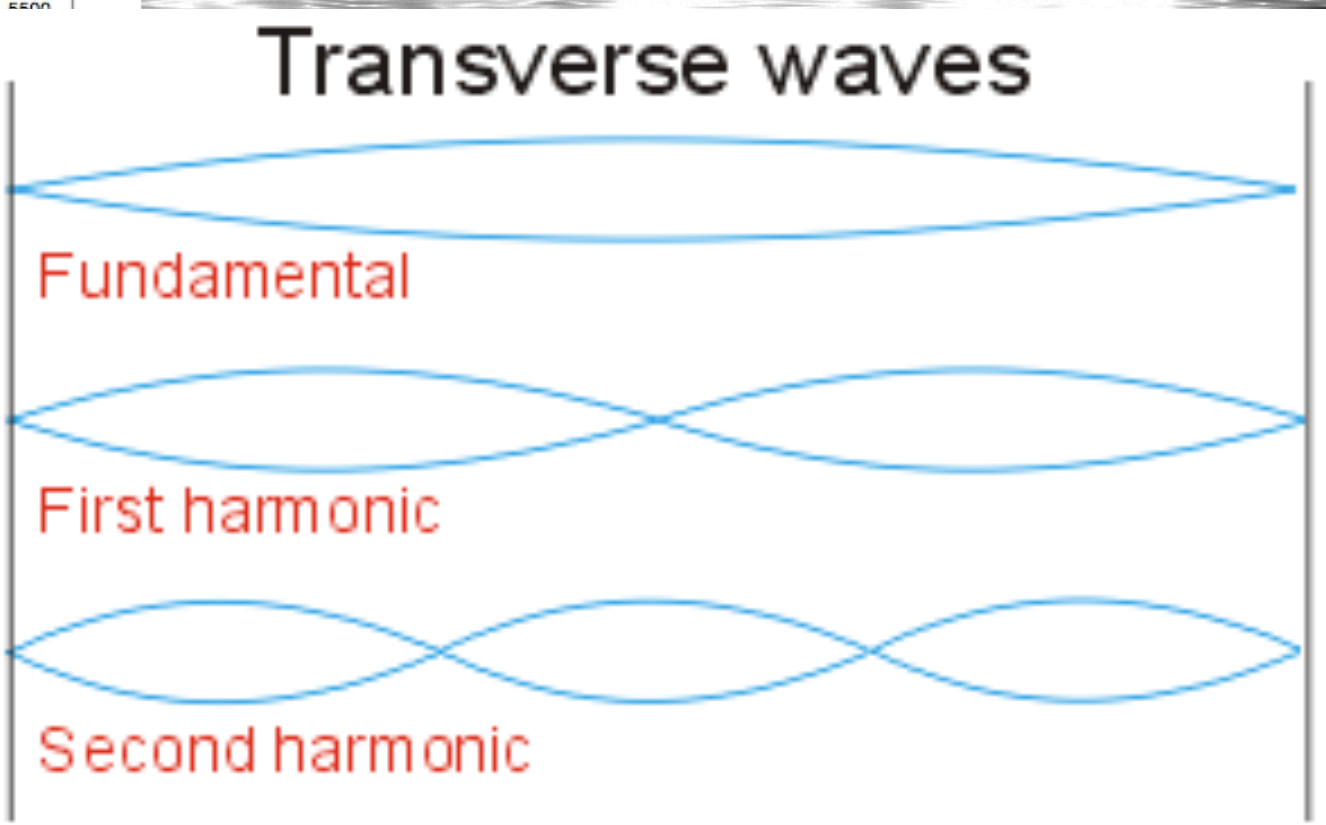


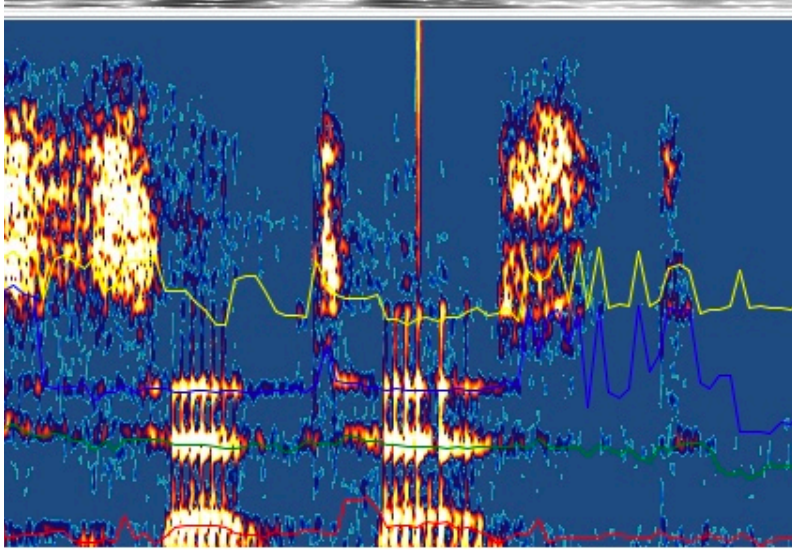
USC Viterbi School of Engineering

Formants, Pitch, Waveform, Spectrum

SCUBA



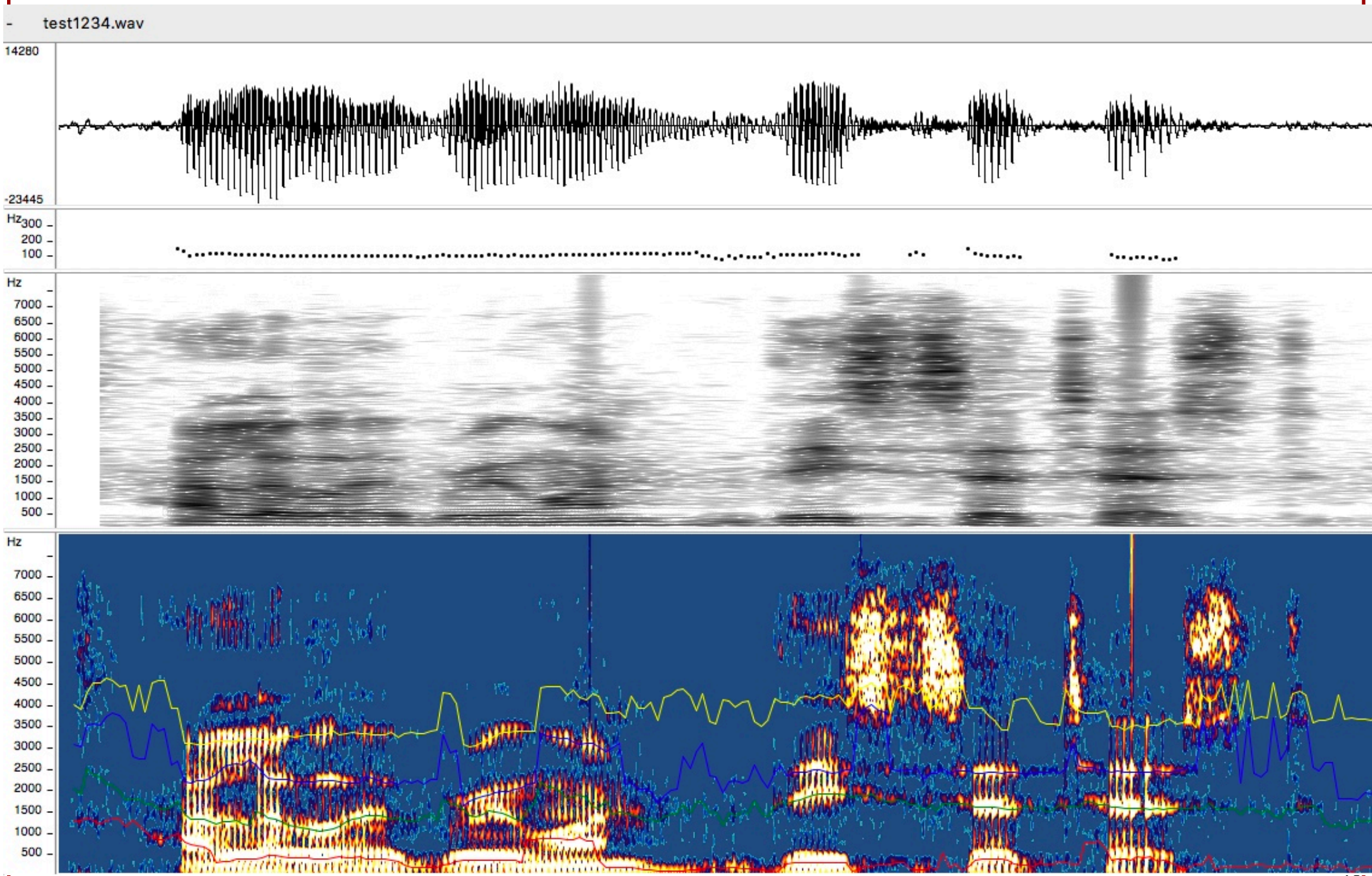


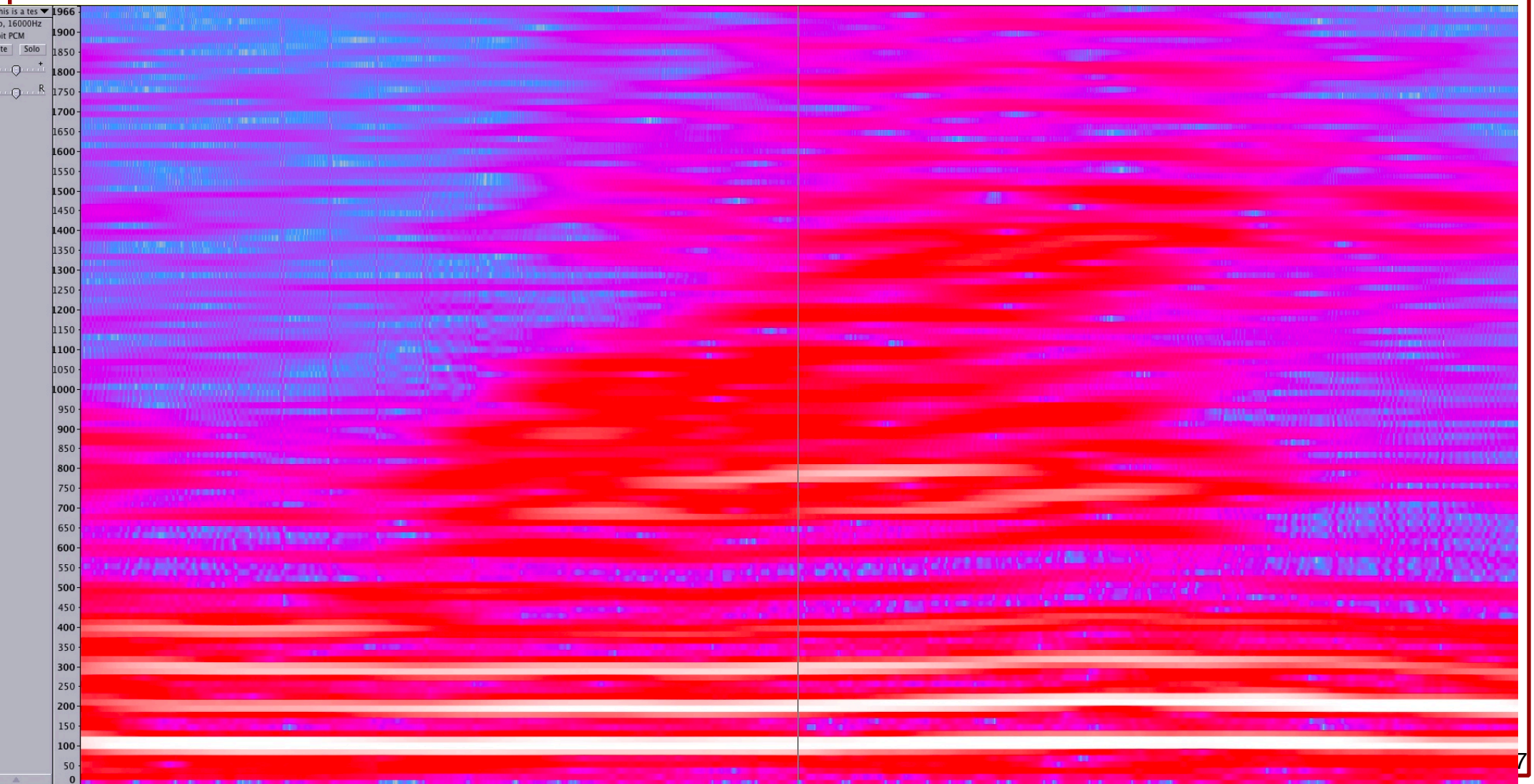


USC Viterbi School of Engineering

Formants, Pitch, Waveform, Spectrum

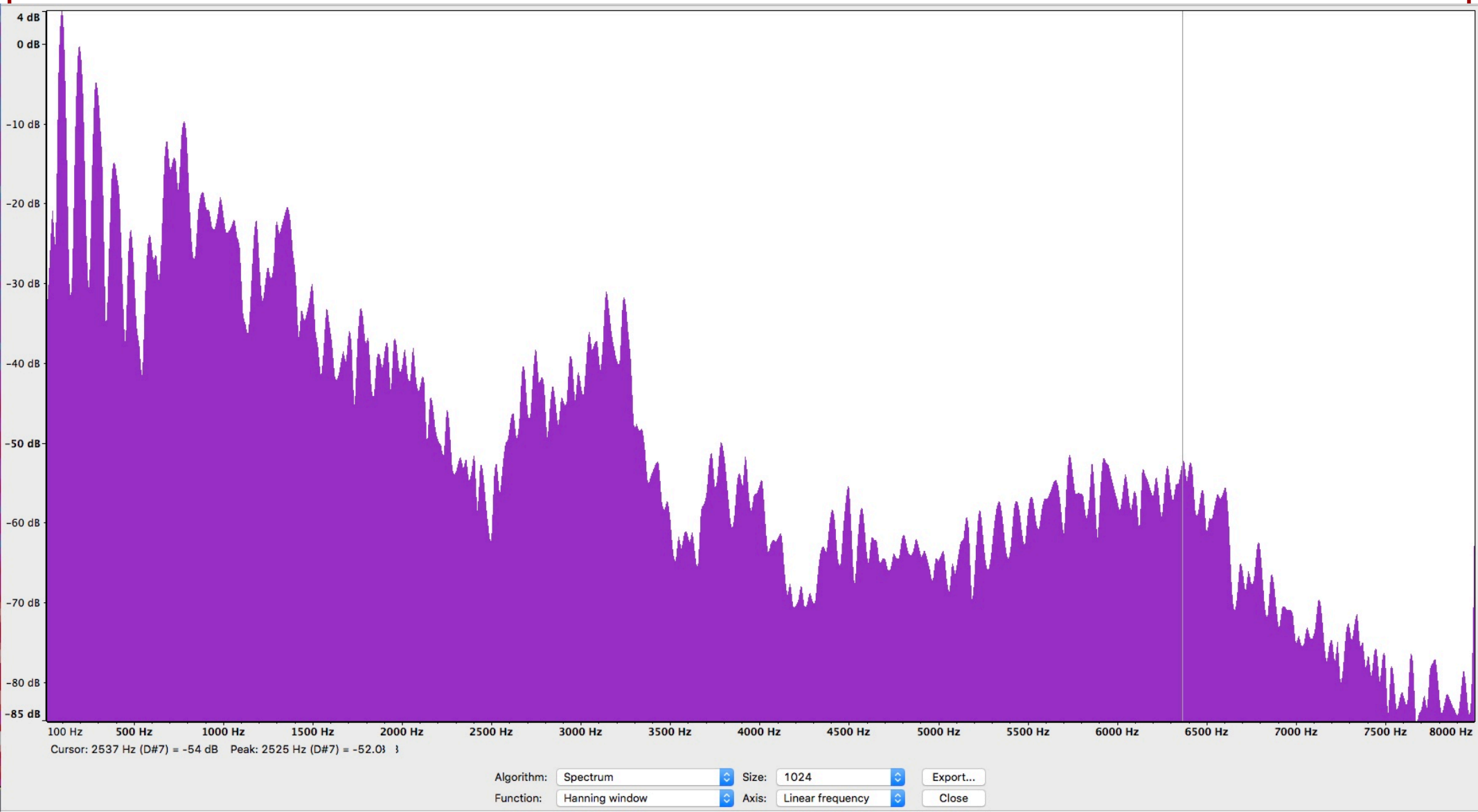
SCUBA




1966
1900
1850
1800
1750
1700
1650
1600
1550
1500
1450
1400
1350
1300
1250
1200
1150
1100
1050
1000
950
900
850
800
750
700
650
600
550
500
450
400
350
300
250
200
150
100
50
0

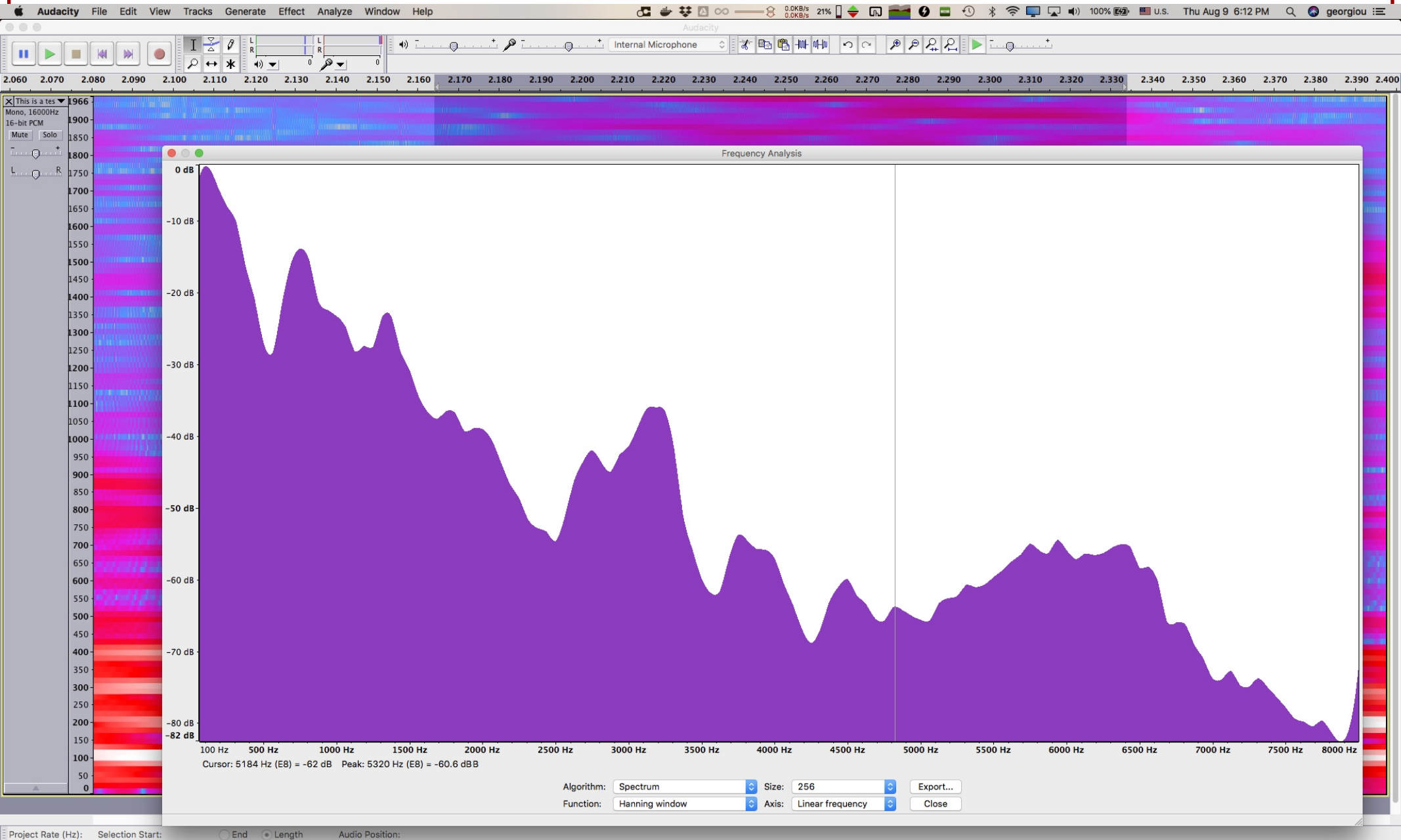
USC Viterbi
School of Engineering

SCUBA



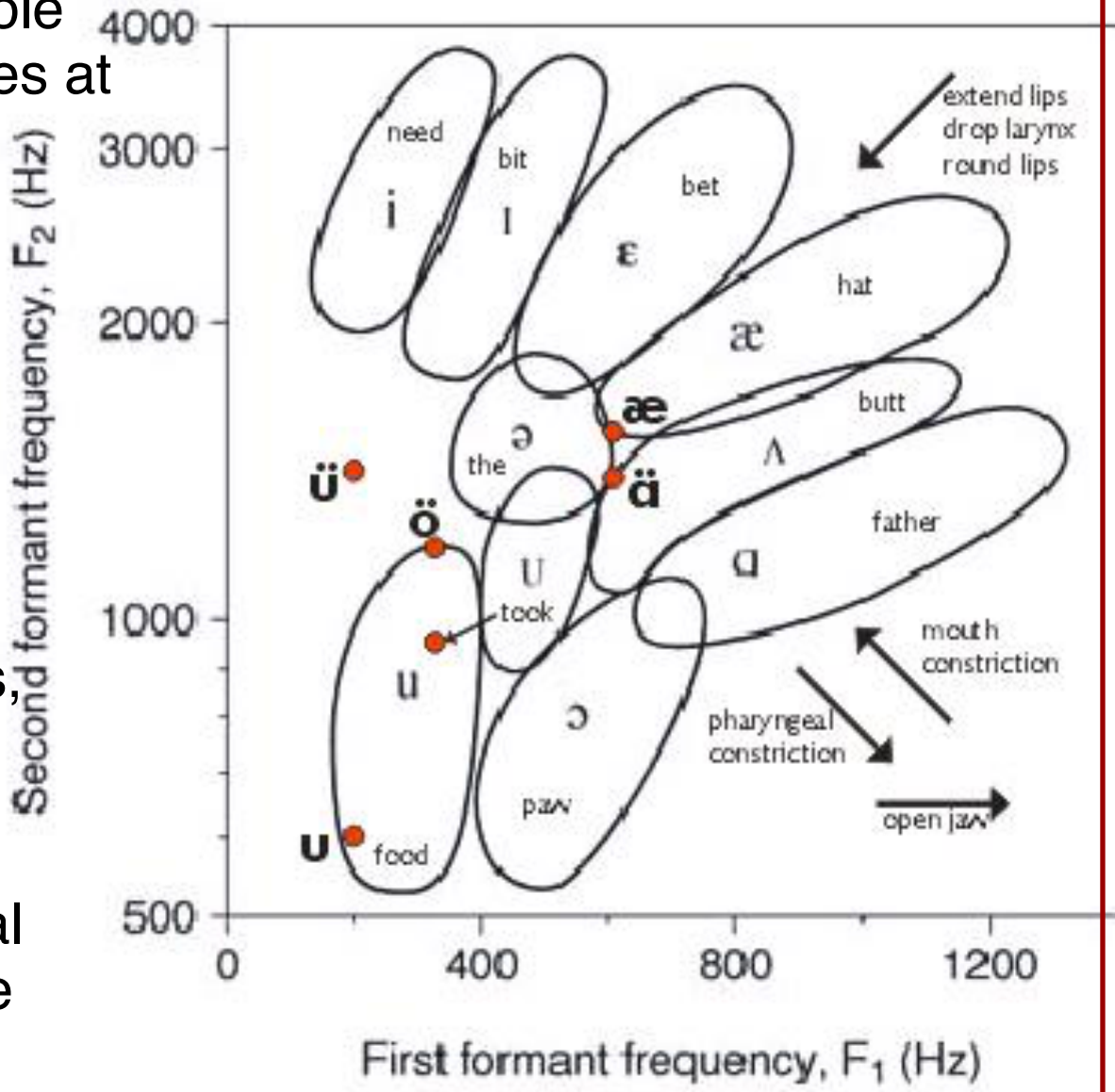
USC Viterbi
School of Engineering

SCUBA



2D formant space

- In general the concatenation of p lossless tubes results in an p-pole system with p/2 conjugated poles at the most.
- These poles are called resonances or formants.
- They occur when a given frequency "gets trapped".
- Relationship between p and F:
  $p = 2 L F / c$
- Example: F=8000Hz, c=340m/s, L=17cm (adult vocal tract)
  $\Rightarrow p=8 \Rightarrow$ 4 formants
- Experimentally shown, the vocal tract transfer function (of a male adult) has about 1 formant per kilohertz



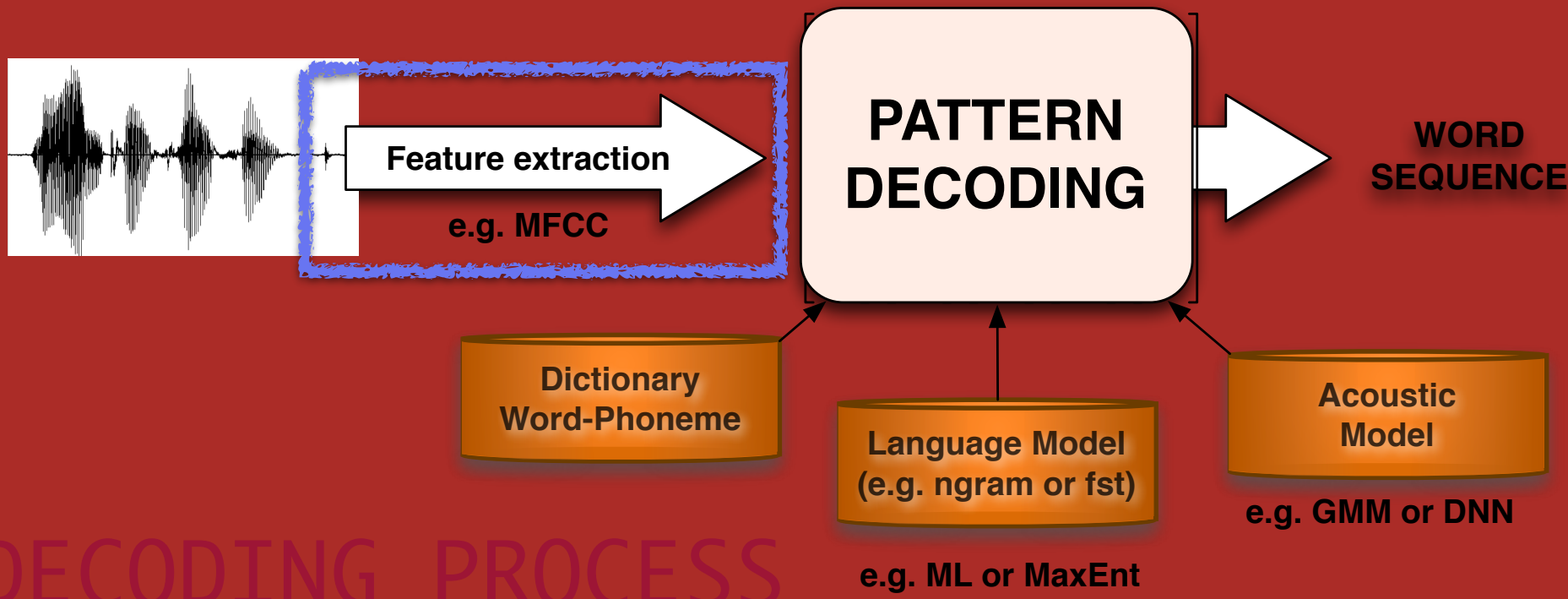
The McGurk Effect

The McGurk Effect

Some Terms

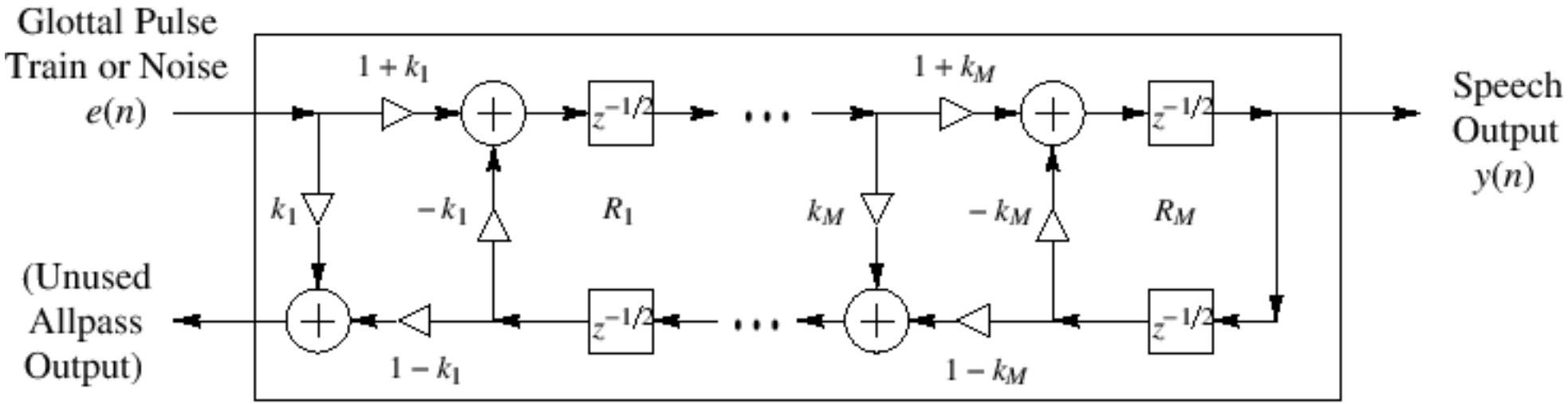
- Formants
- Pitch or Fundamental Frequency f_0
  - *Strictly pitch is the perceived and f_0 the measured*
- Amplitude
  - Can be represented as power (= integral of the squared signal) or peak-to-peak distances
  - Useful for detecting speech vs silence, also syllables, phrase boundaries, prosody
- Periodicity
  - Homogeneous periodic signal indicates voiced sounds
  - White noise indicated unvoiced sounds
- Zero-crossing rate
  - Number of times the signal crosses zero per unit time
  - Can help distinguish some weak sounds from silence

Features: LPC



DECODING PROCESS

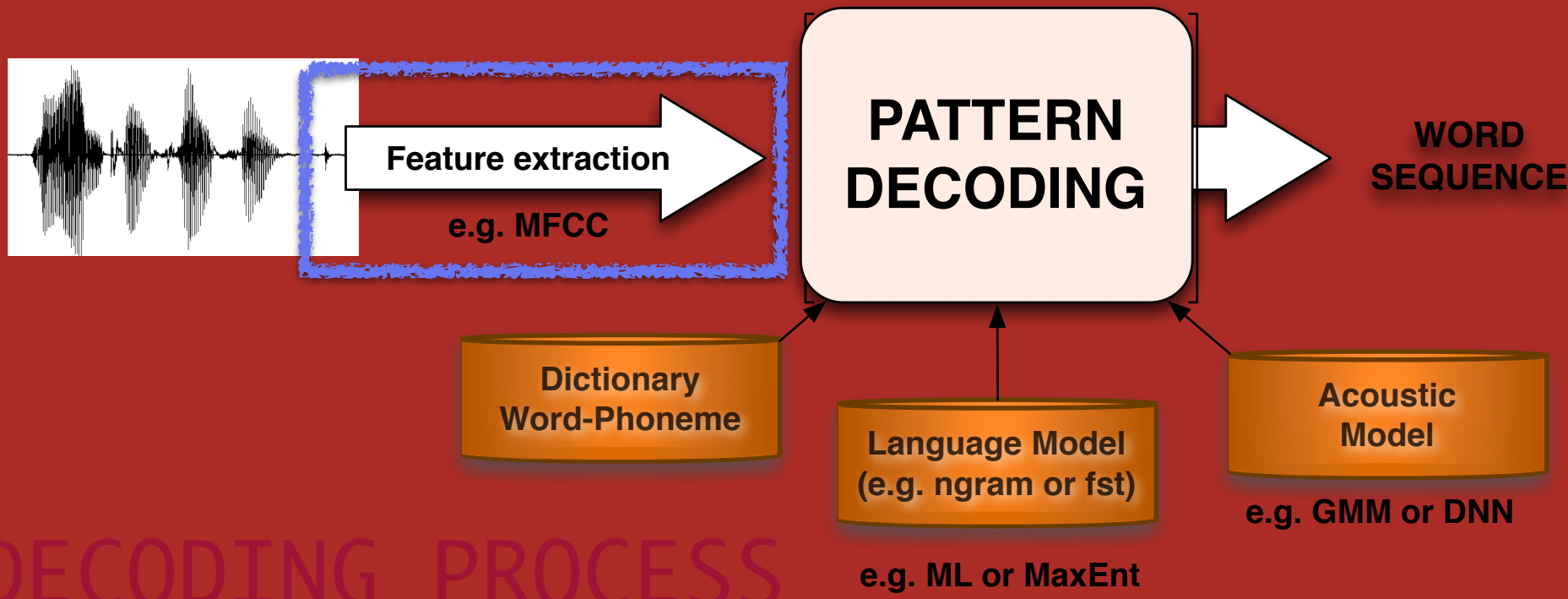
Linear Prediction Coding



Kelly-Lochbaum Vocal Tract Model (Piecewise Cylindrical)

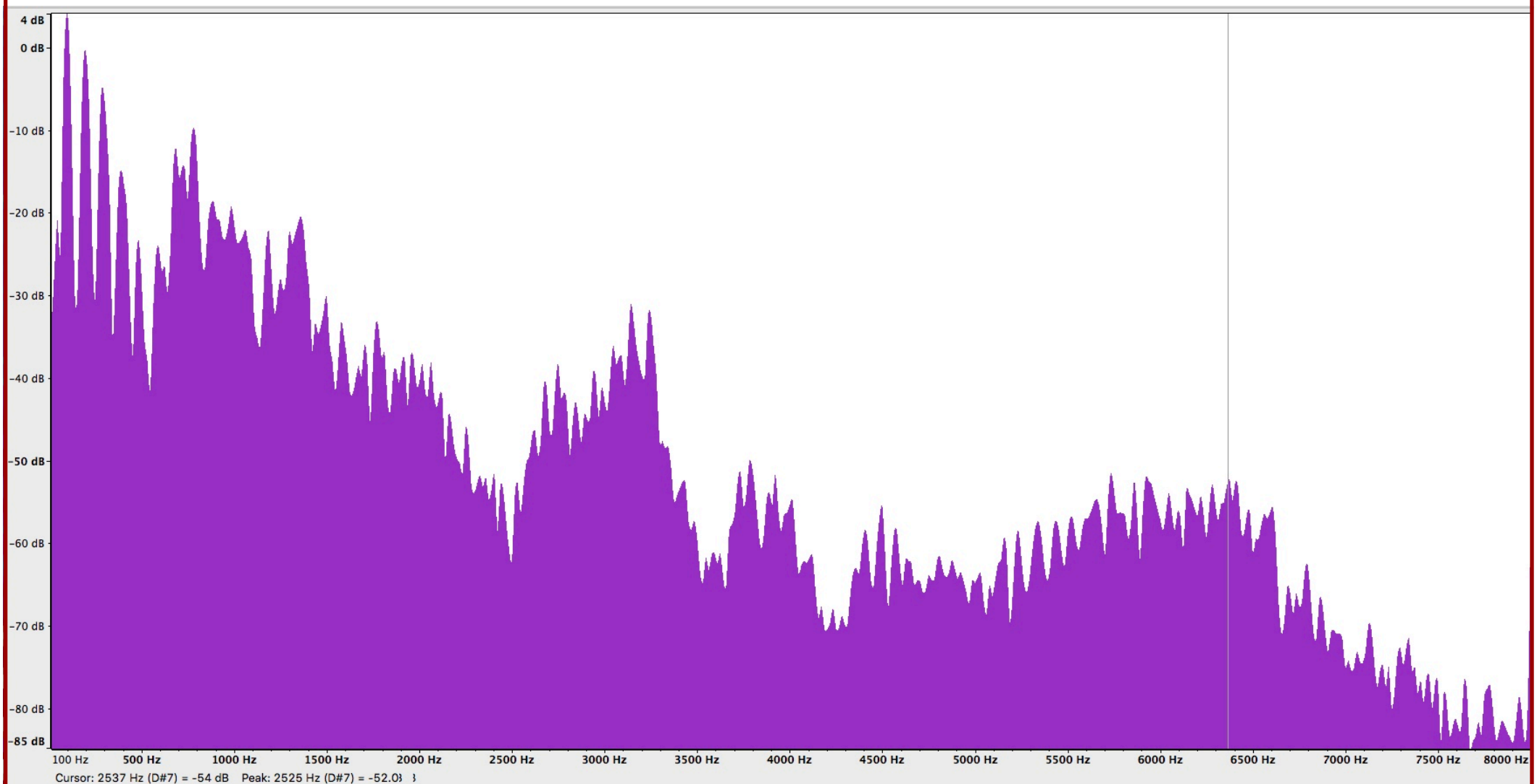
- Use the coefficients of this as a feature
- "Good" for compression (40 years ago)
- 1984 LPC-10 government standard
  - with a lot of tweaks. Not like above
- Has been used for early speech recognition work

Features: MFCC



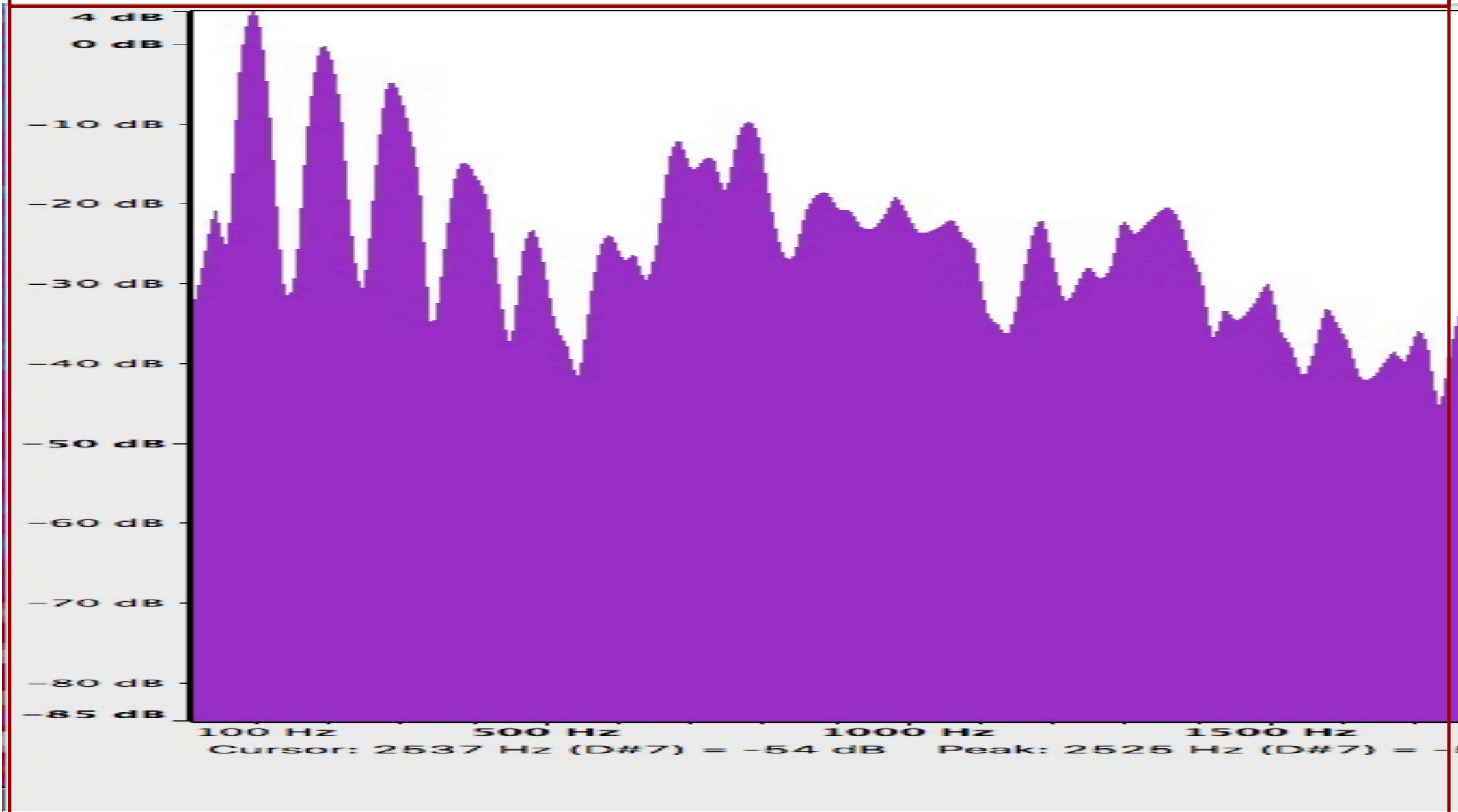
DECODING PROCESS

From speech production to features



Voiced Segment

Pitch: spacing between peaks



Voiced Segment

From speech production to features

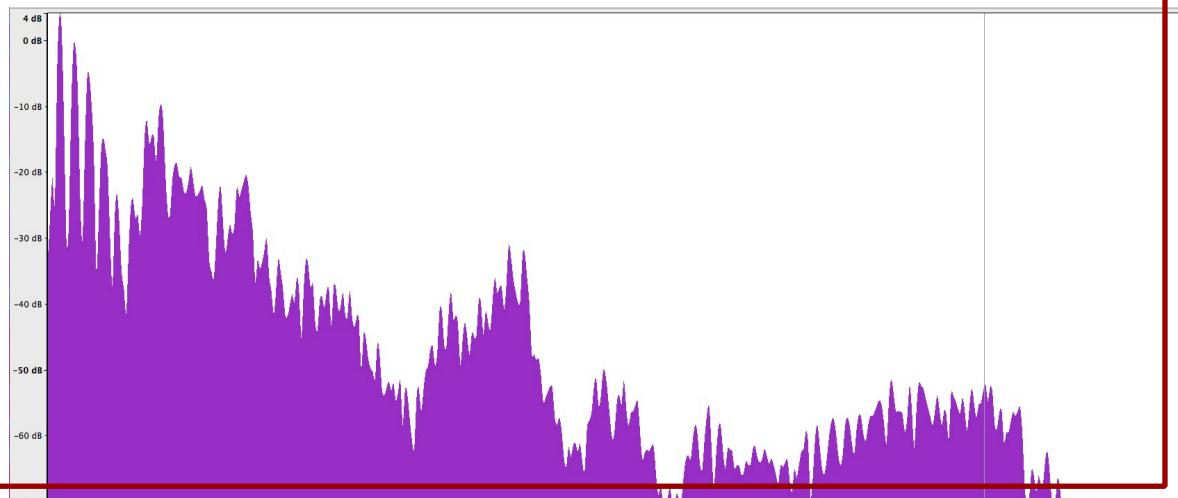
- Observe:
  - f_0 is a multiplicative signal on top of the rest…
  - The spectral envelope that also encodes f_1, f_2, f_3…
  - X (overall) = S (spectral envelope) ⁎ E (Excitation)
  - log(X) = log(S) + log (E)
  - If we transform (ifft) then freqs separate
    E (high freq): Pitch/excitation (emotions, Chinese…)
    S (low freq): What we mostly want

Voiced Segment

From speech production to features

- Observe:
  - f_0 is a multiplicative signal on top of the rest…
  - The spectral envelope that also encodes f_1, f_2, f_3…
  - X (overall) = S (spectral envelope) ⁎ E (Excitation)
  - log(X) = log(S) + log (E)
  - If we transform (ifft) then freqs separate
    E (high freq): Pitch/excitation (emotions, Chinese…)
    S (low freq): What we mostly want

Voiced Segment

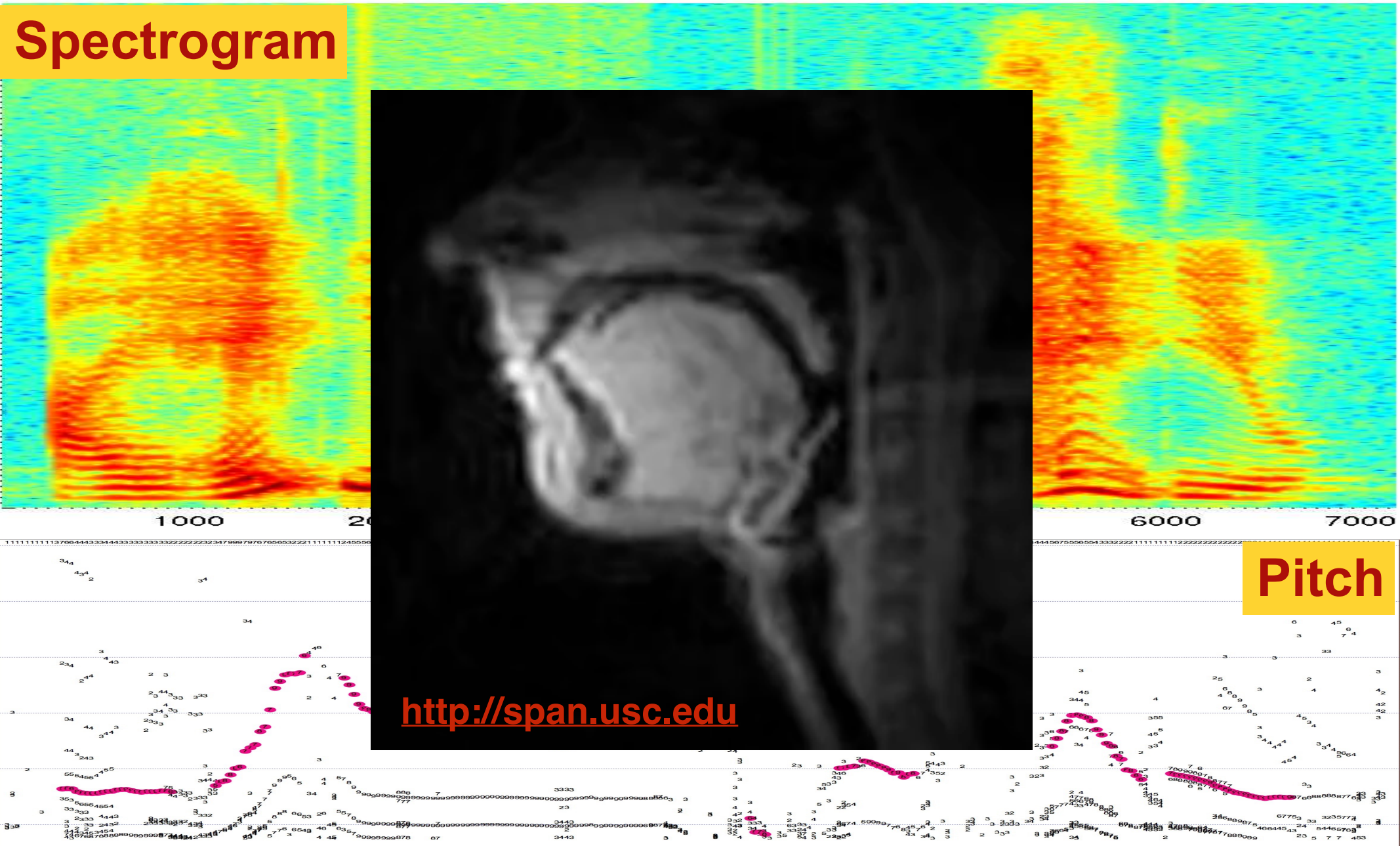


USC Viterbi School of Engineering

Pitch or spectral features

SCUBA

Spectrogram



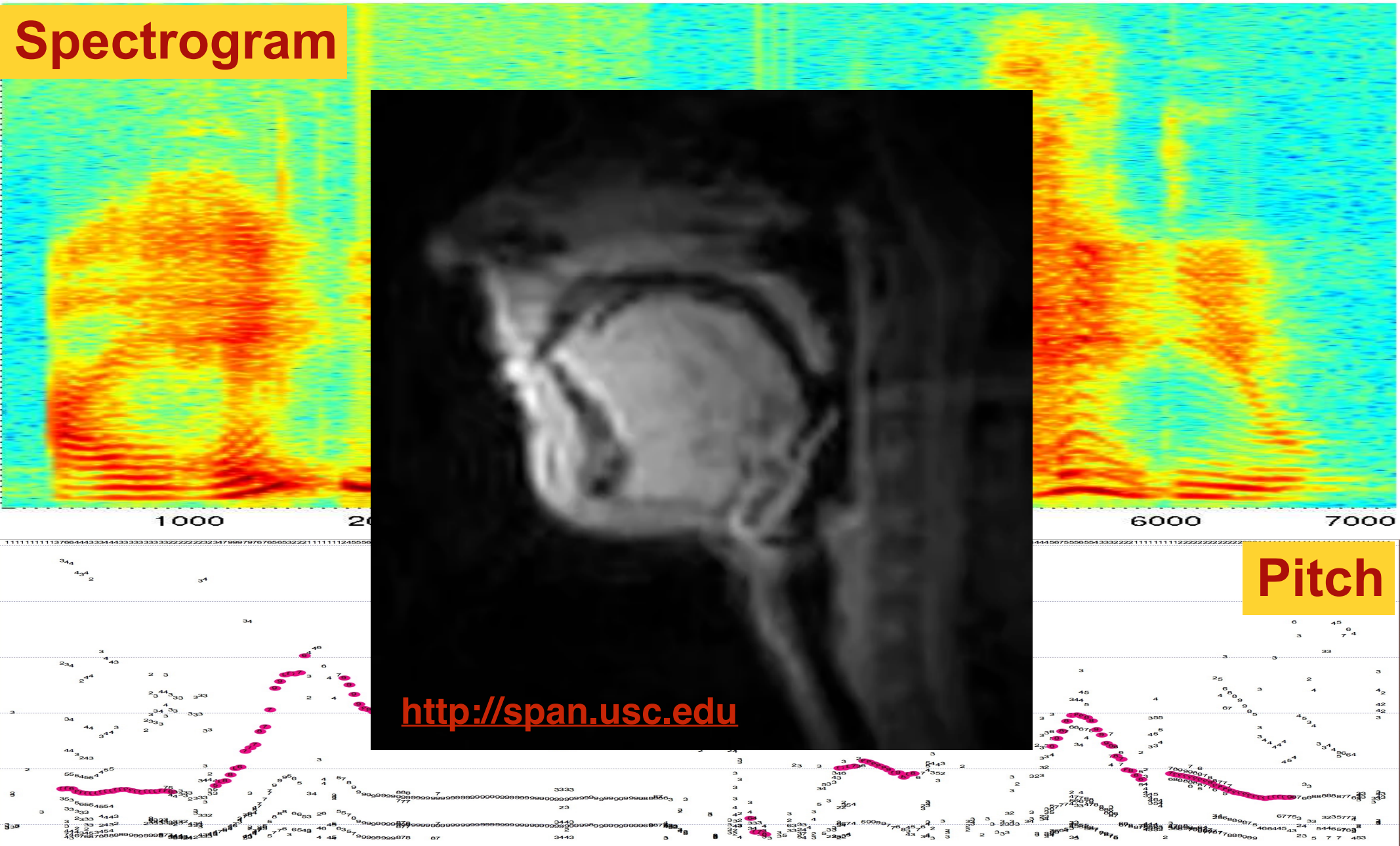
Pitch

USC Viterbi School of Engineering

Pitch or spectral features

SCUBA

Spectrogram



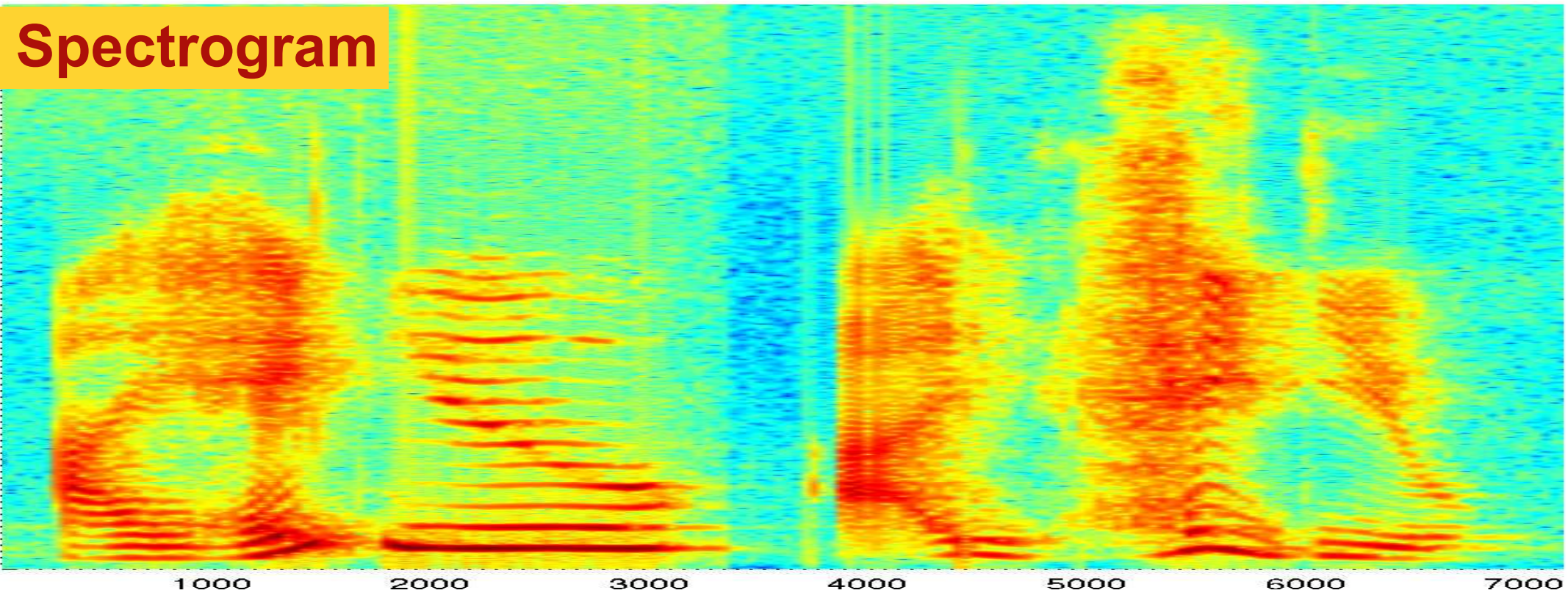
Pitch

USC Viterbi School of Engineering

Pitch or spectral features$_2$

SCUBA

Spectrogram



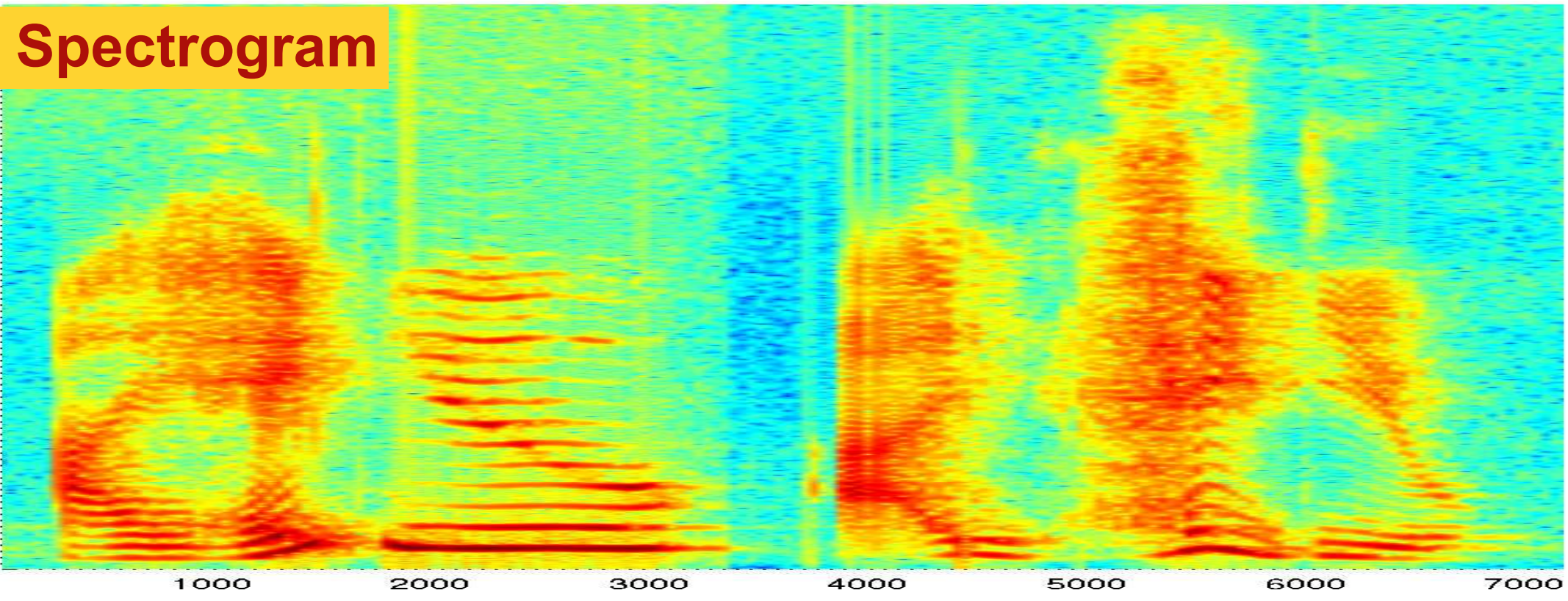
Pitch

USC Viterbi School of Engineering

Pitch or spectral features$_2$

SCUBA

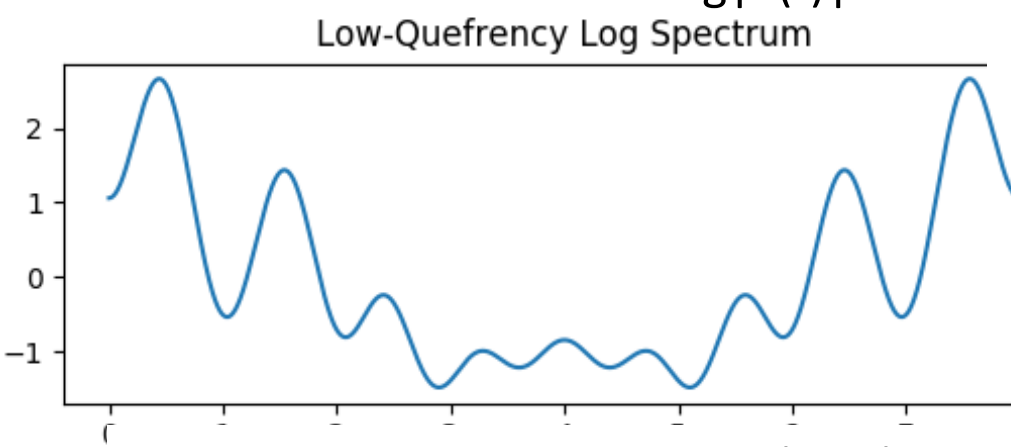
Spectrogram



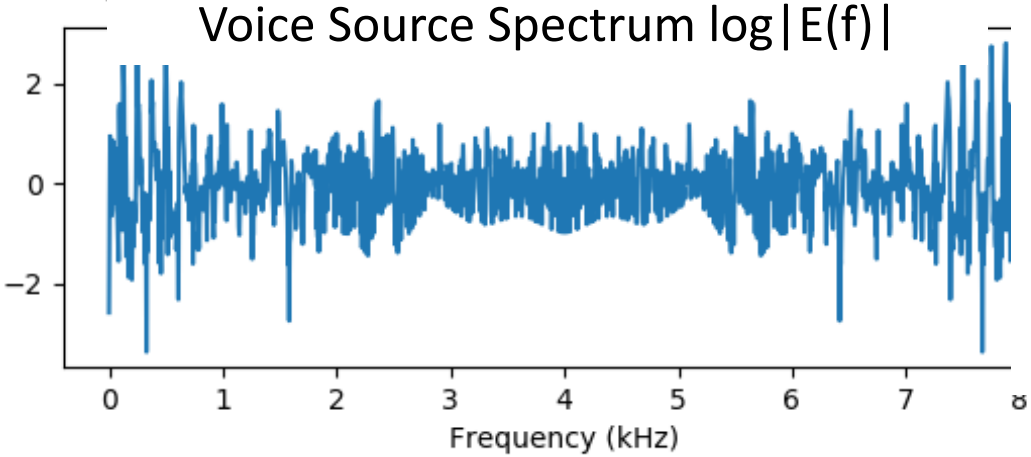
Pitch

The Source-Filter Model

Transfer Function log|S(f)|

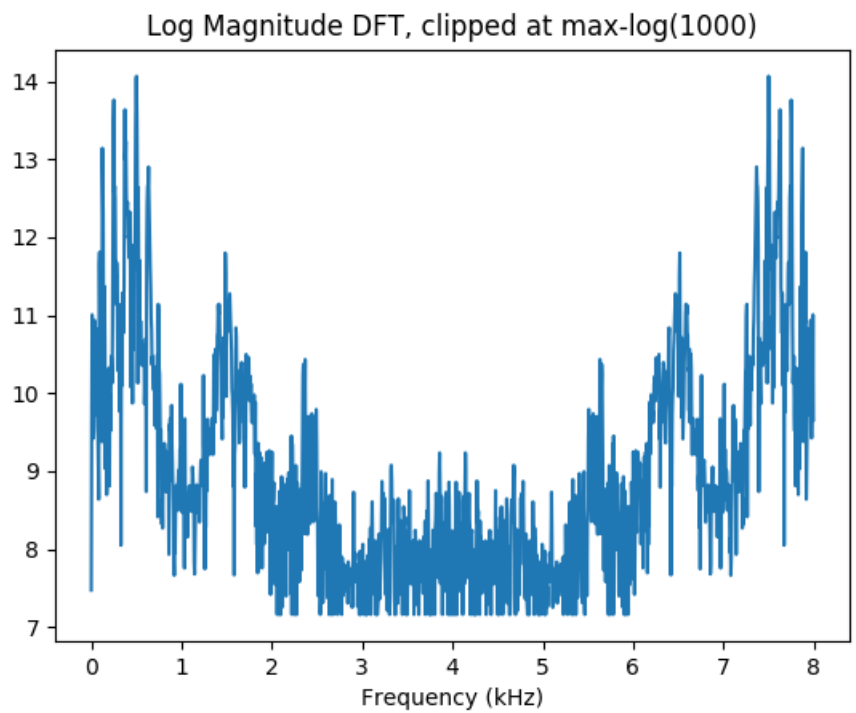


Voice Source Spectrum log|E(f)|



Speech Spectrum

log|X(f)|=log|S(f)|+log|E(f)|



Cepstrum

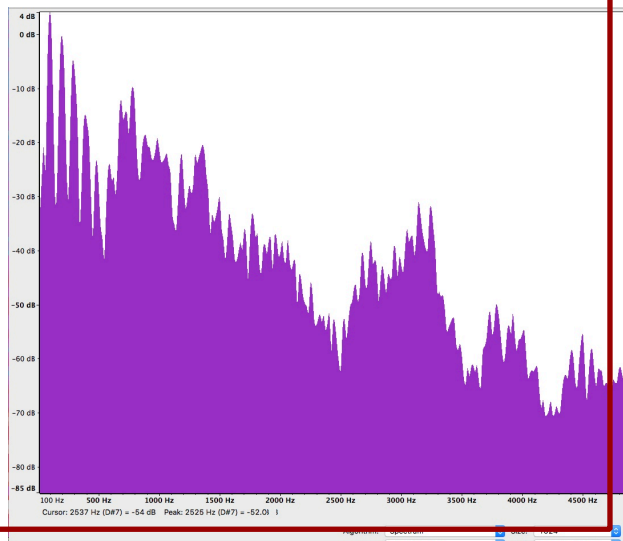
- X(k) = S (k) ⁎ E (k)
- ifft(log()) converts this to the cepstrum domain
- x(k)=ifft(log(X)) is called the cepstrum

- i.e. (time domain) → **fft** →
  (freq domain) → **log ()** →
  (log freq domain) → **ifft ()** →
  (cepstrum domain) → ***Cepstrum***

Cepstrum

- X(k) = S (k) ⁎ E (k)
- ifft(log()) converts this to the cepstrum domain
- x(k)=ifft(log(X)) is called the cepstrum

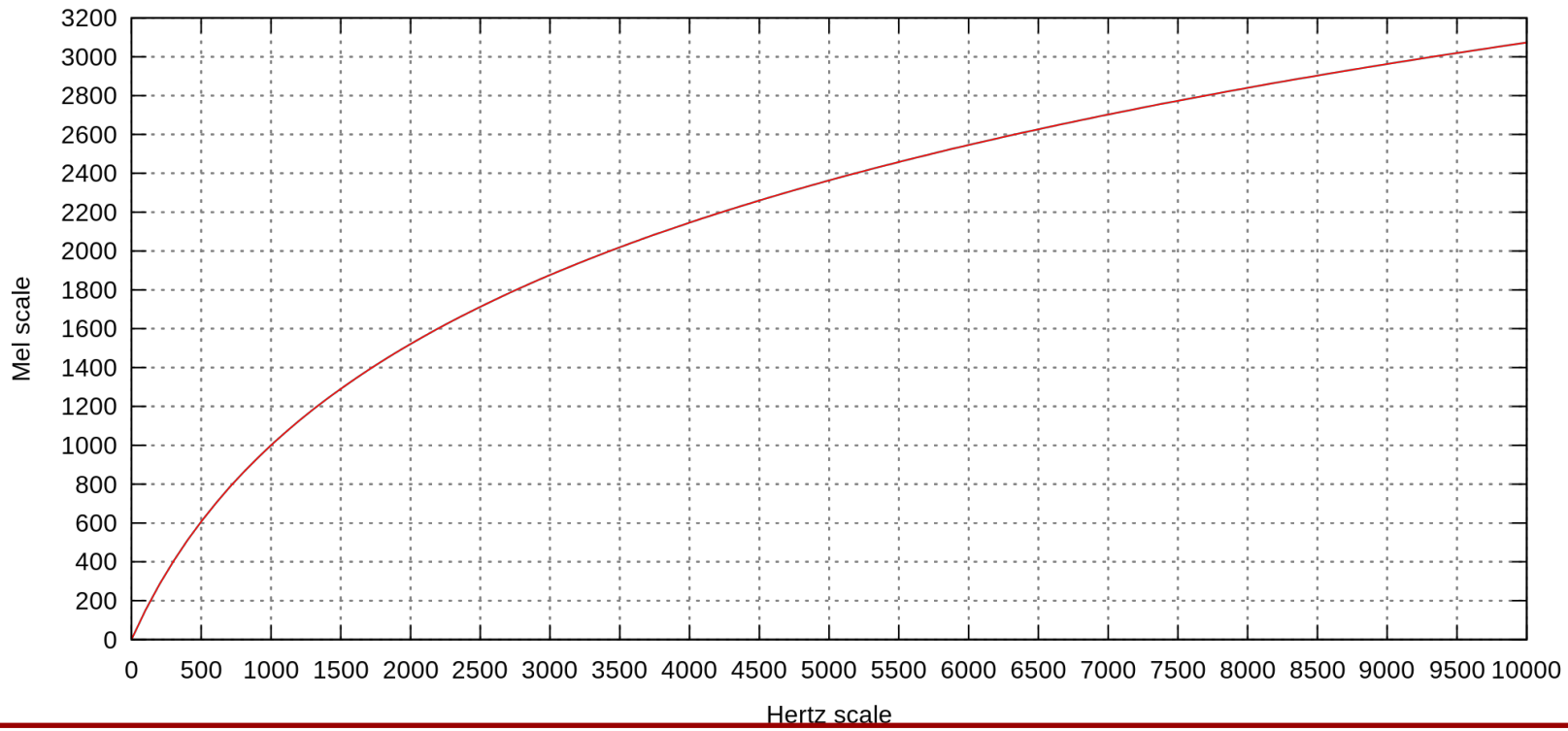
- i.e. (time domain) → **fft** →
  (freq domain) → **log ()** →
  (log freq domain) → **ifft ()** →
  (cepstrum domain) → ***Cepstrum***



Mel Scale

- Mel Scale:
- People don't judge freq as linear
- Asking people to judge the distance between two pairs of sounds results in the Mel Scale

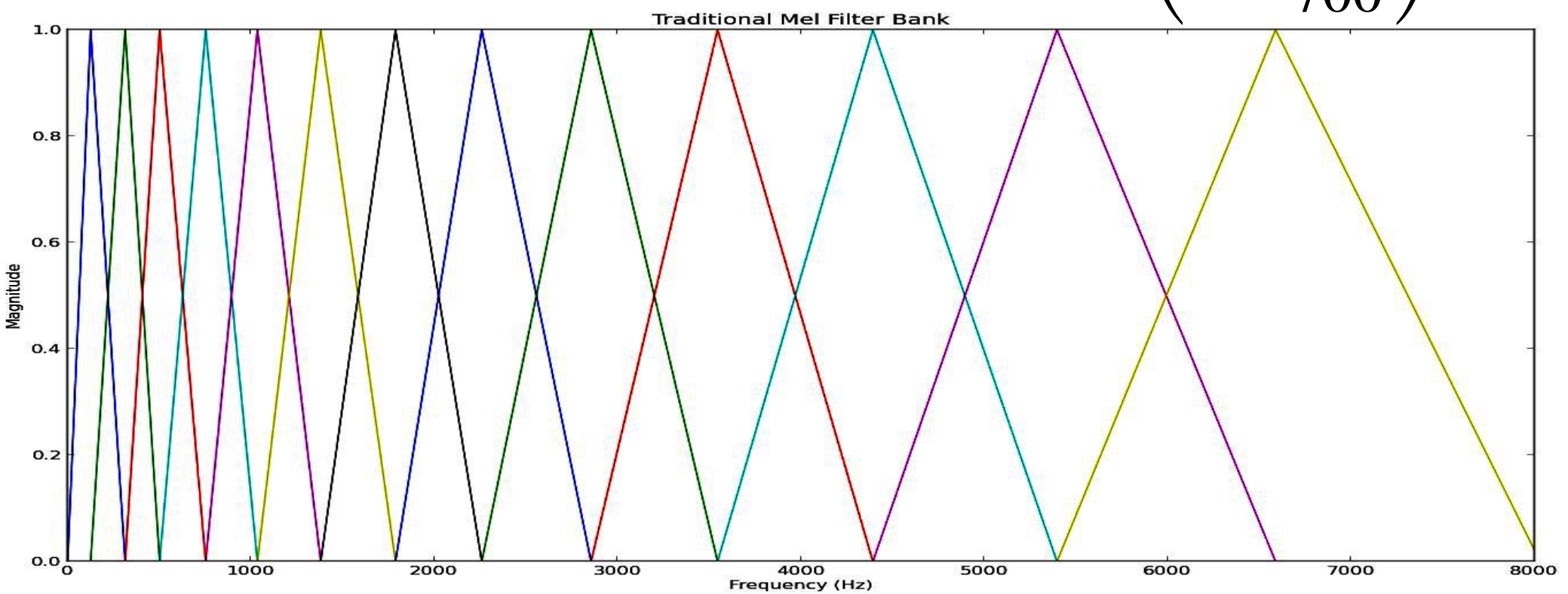
$$m = 2595 \log_{10}\left(1 + \frac{f}{700}\right)$$



Mel Scale

- Mel Scale:
- People don't judge freq as linear
- Asking people to judge the distance between two pairs of sounds results in the Mel Scale

$$m = 2595 \log_{10}\left(1 + \frac{f}{700}\right)$$



Mel in practice

Bin ***Frequency*** ***Mel*** ***Mel Bin***

0 ← 0 → 2595 log(1+0/700) → 0

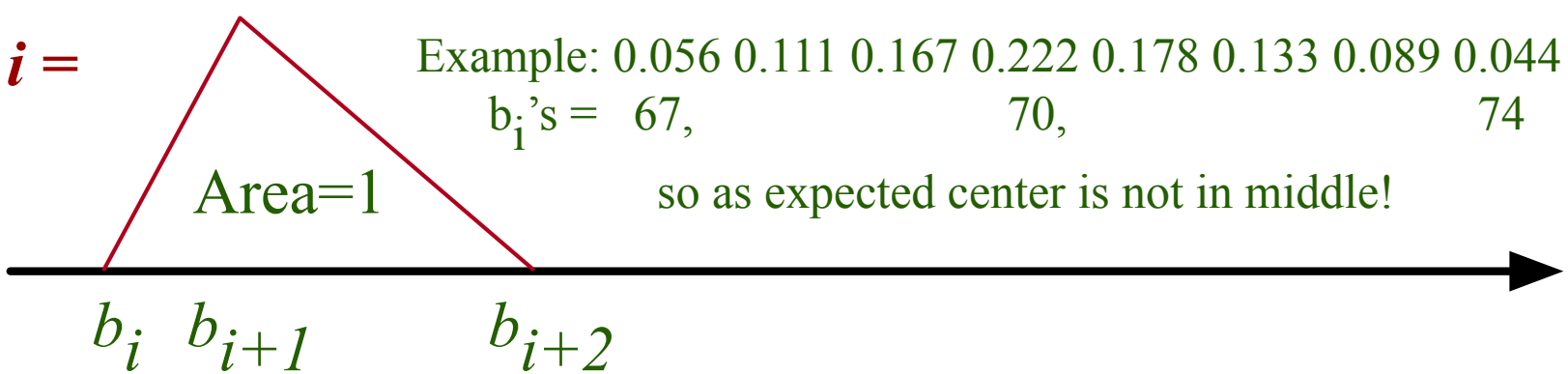
↓

b_n ← f_n ← Divide linearly

↑

255 ← 8kHz → 2595 log(1+8000/700) → 42

Mel i =



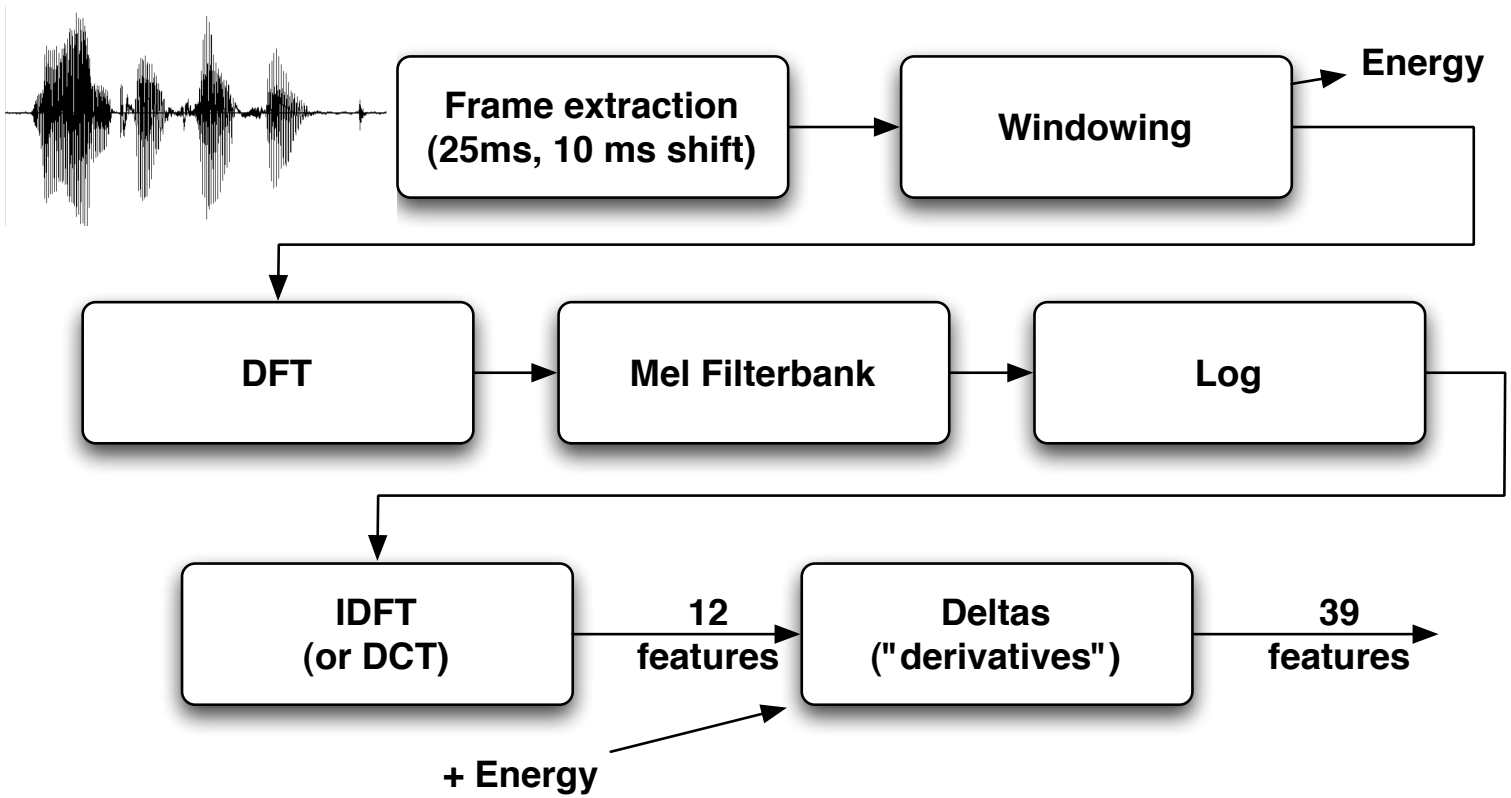
Example: 0.056 0.111 0.167 0.222 0.178 0.133 0.089 0.044

b_i's = 67, 70, 74

so as expected center is not in middle!

Mel Freq. Cepstral Coefficients (MFCC)

- i.e. (time domain) → **fft** →
  (freq domain) → **Mel Filterbank**
  (Mel freq domain) → **log ()** →
  (log freq domain) → **ifft ()** →
  (Mel Freq. Cepstrum domain) → ***MFCC***

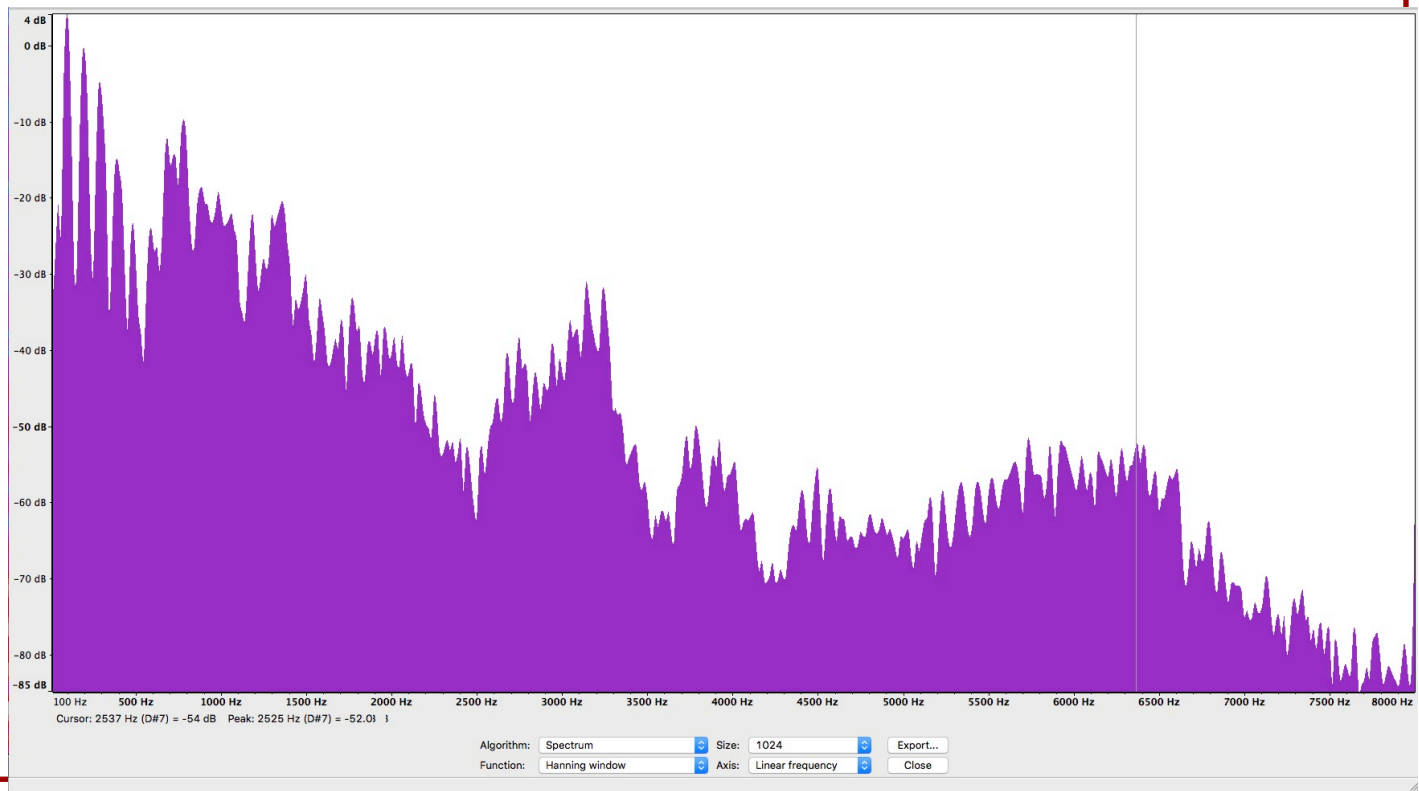


Pre-emphasis

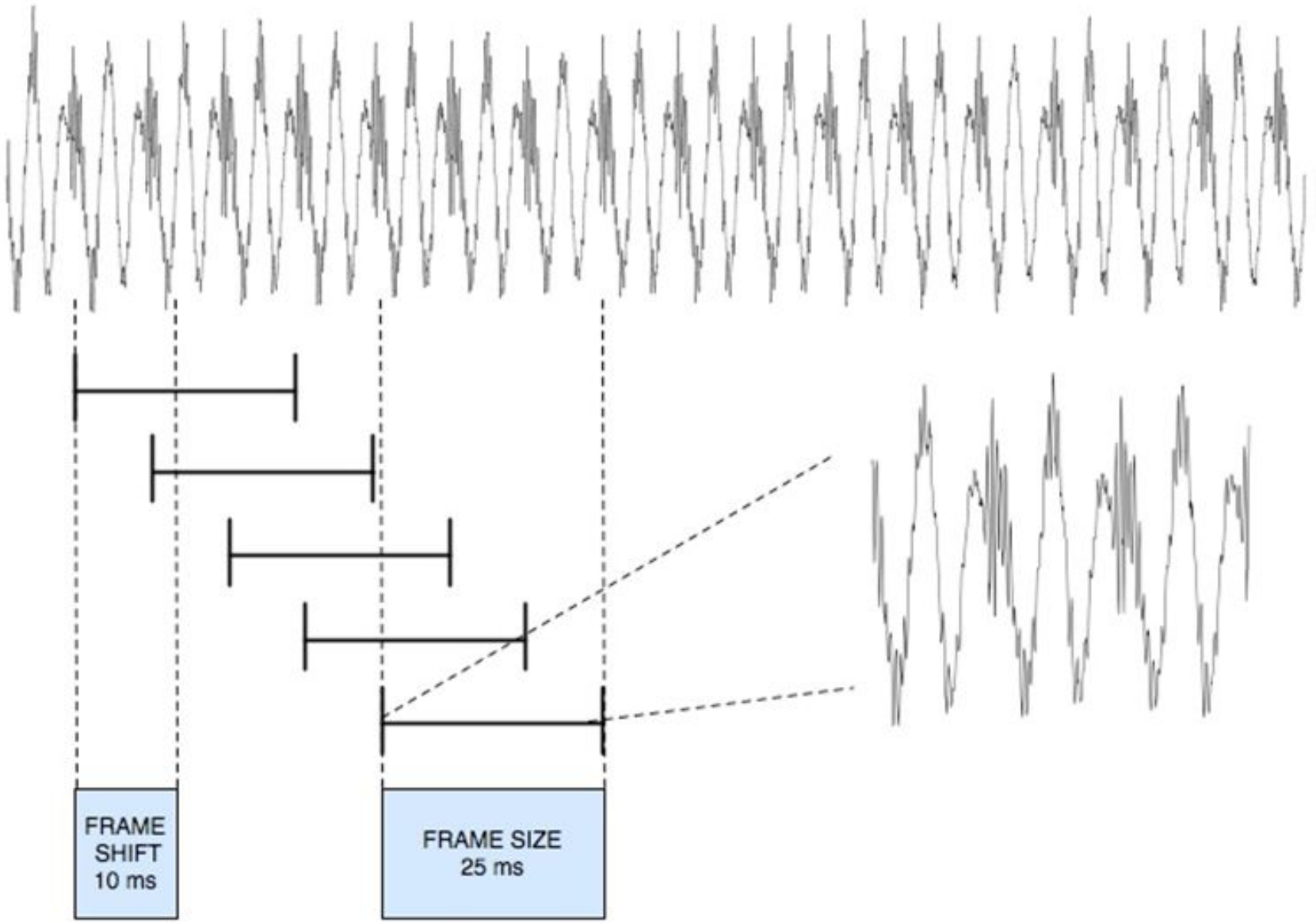
- Side note:
  - we usually (always) pre-emphasize the signal
  - y(n)=x(n)- 0.97x(n-1)
  - as we saw we need to boost higher frequencies

Pre-emphasis

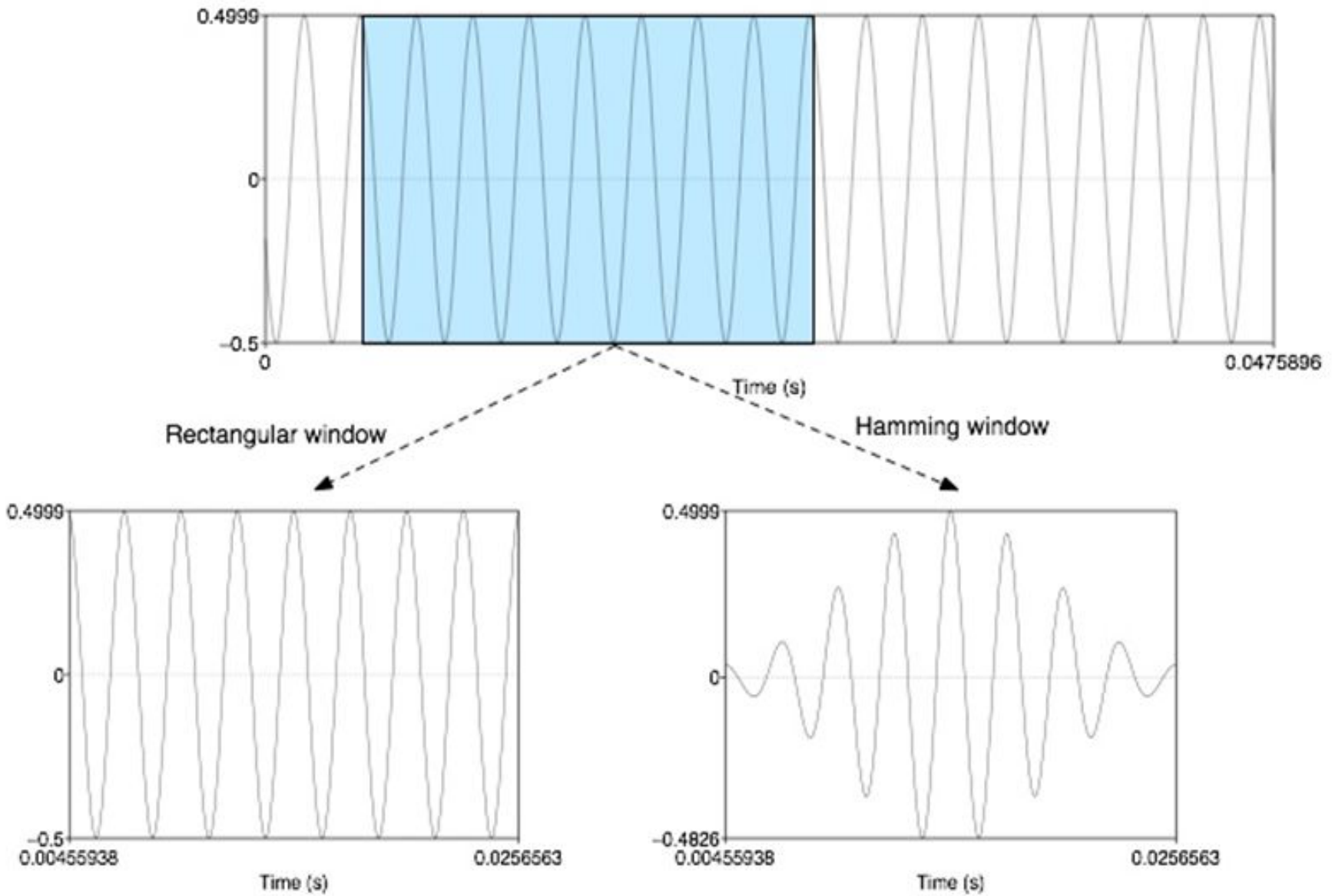
- Side note:
  - we usually (always) pre-emphasize the signal
  - $y(n)=x(n)- 0.97x(n-1)$
  - as we saw we need to boost higher frequencies



Overlap



Windowing


0.4999
0
−0.5
0
0.0475896
Time (s)
Rectangular window
Hamming window
0.4999
0
−0.5
0.00455938
0.0256563
Time (s)
0.4999
0
−0.4826
0.00455938
0.0256563
Time (s)

Main Types of speech sounds

- Voiced (e.g. vowels):
  - Vocal chord vibrates
  - Pitch is defined
- Unvoiced (consonants)
  - No vocal chord vibration
  - Significant variability
  - Tougher to model

- Understanding these types (and ones on next slides)
  - Can help in good feature extraction
  - Can help in 'tying' (ASR classes)

Based on articulation

- https://www.phon.ucl.ac.uk/courses/spsci/iss/week6.php

Place	Description
Bilabial	both lips
Labiodental	lower lip against upper teeth
Dental	tongue tip against upper teeth
Alveolar	tongue tip against teeth ridge
Alveolar-lateral	tongue tip against teeth ridge but with sides lowered
Alveolar-retroflex	tongue tip curled back near teeth ridge
Palato-alveolar	tongue tip slightly retracted from teeth ridge
Palatal	tongue blade against hard palate
Velar	back of tongue against soft palate
Glottal	vocal fold closure in larynx

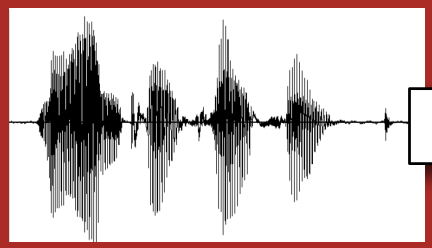
USC Viterbi School of Engineering

Based on “narrowing"

SCUBA

Manner	Description
Approximant	sounds do not cause complete obstructions of the vocal tract, they are just narrowings of the tract in different ways at different positions. They are all voiced.
Plosive	sounds have a complete obstruction at some place. These cause an interruption to the air-flow and to the passage of sound from the larynx. For voiceless plosives there is a simultaneous glottal opening gesture as well as an oral closure.
Fricative	sounds involve a severe narrowing of the air path at some place. The narrowing causes the air flow to become turbulent and to create noise. In voiced fricatives, this turbulent noise is added to the sound of phonation from the larynx.
Affricate	sounds are like a combination of plosive and fricative: a plosive like oral closure followed by a fricative release.
Nasal	sounds have oral closures like the plosives, but these are made in combination with a lowered soft palate that allows air flow out through the nose. The continuation of flow means that voicing can be maintained.

- Speech Production “demo”
  - https://youtu.be/J3TwTb-T044



AM

Feature extraction

e.g. MFCC

PATTERN DECODING

WORD SEQUENCE

Dictionary Word-Phoneme

Language Model (e.g. ngram or fst)

e.g. ML or MaxEnt

Acoustic Model

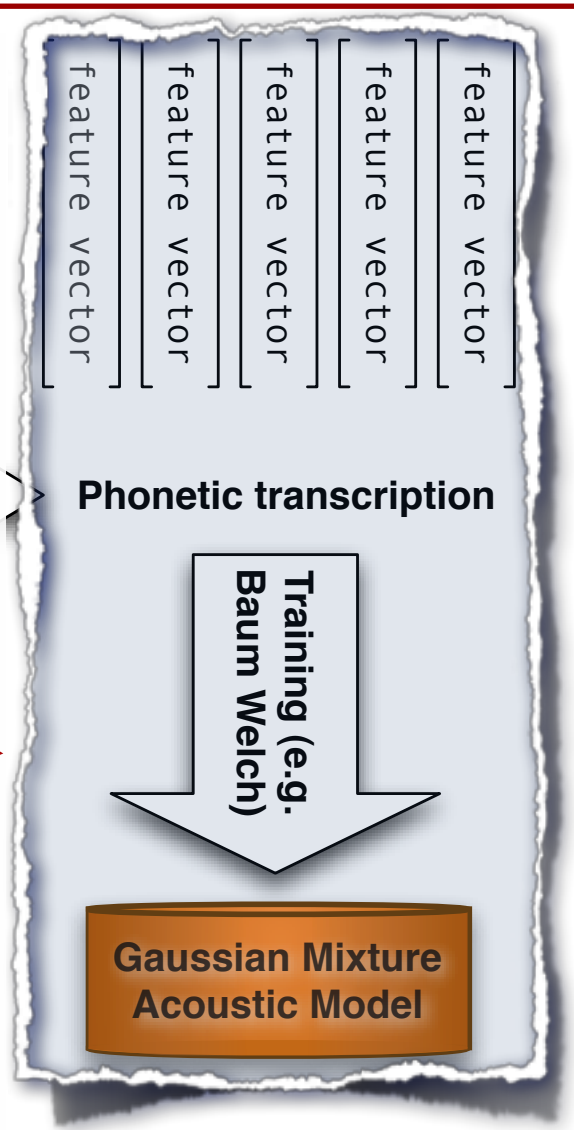
e.g. GMM or DNN

DECODING PROCESS

ASR Training: Acoustic Model

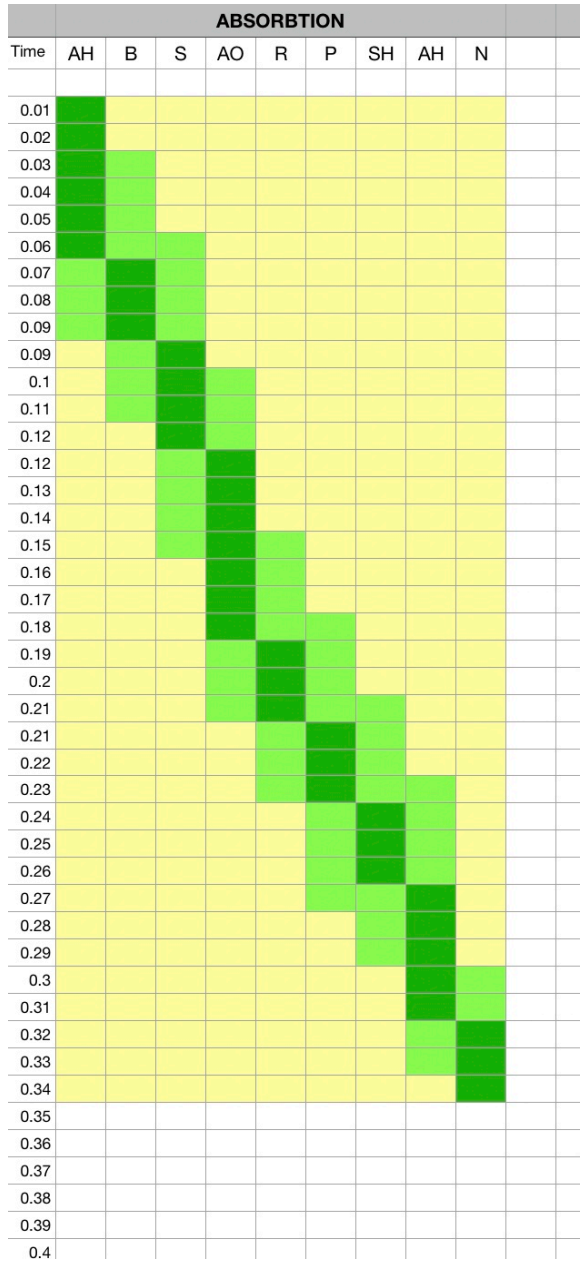
- Gives a prior on what speech sounds like
- Needs to match the domain where ASR will be used

- Usual problem: training data is mismatched with domain data



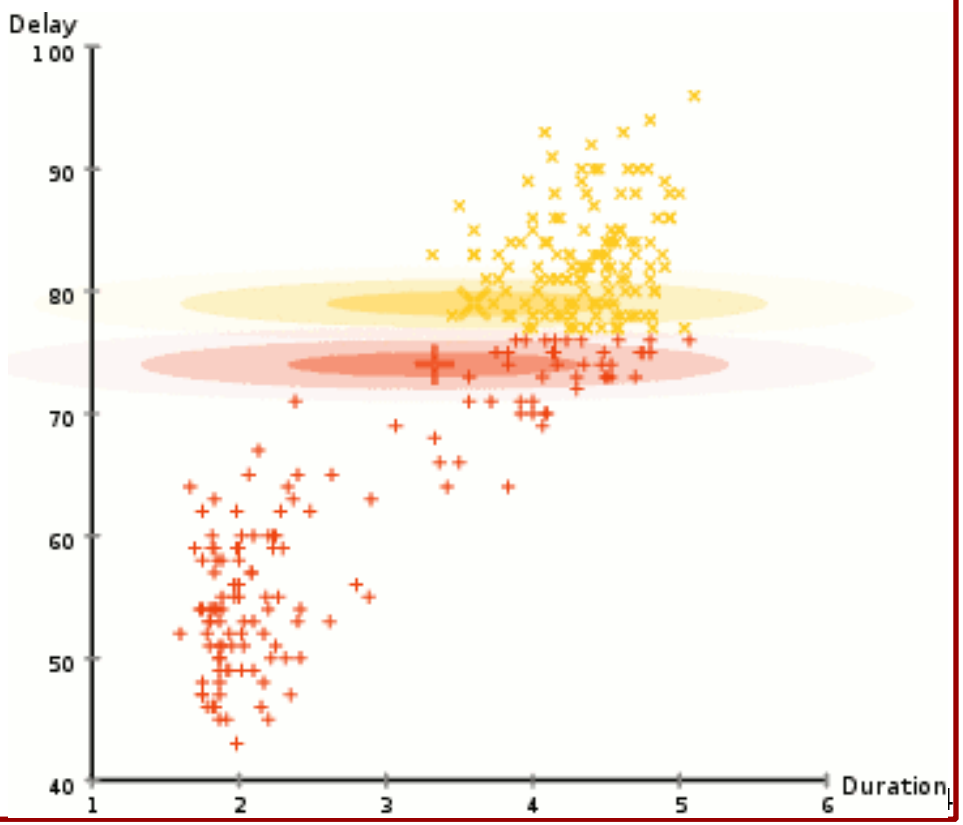
GMM computation

- For each
  - time window
  - Extract feature,
  - assume a phoneme (initially flat init)
  - find the GMM
    - Do EM to find GMM
  - *Use the GMM to decode again and*
  - *find the most likely path*
  - *Use the newly assigned phoneme for the feature*
  - *Find the new GMM*
    - *Do EM to find the GMM*
  - Iterate green part



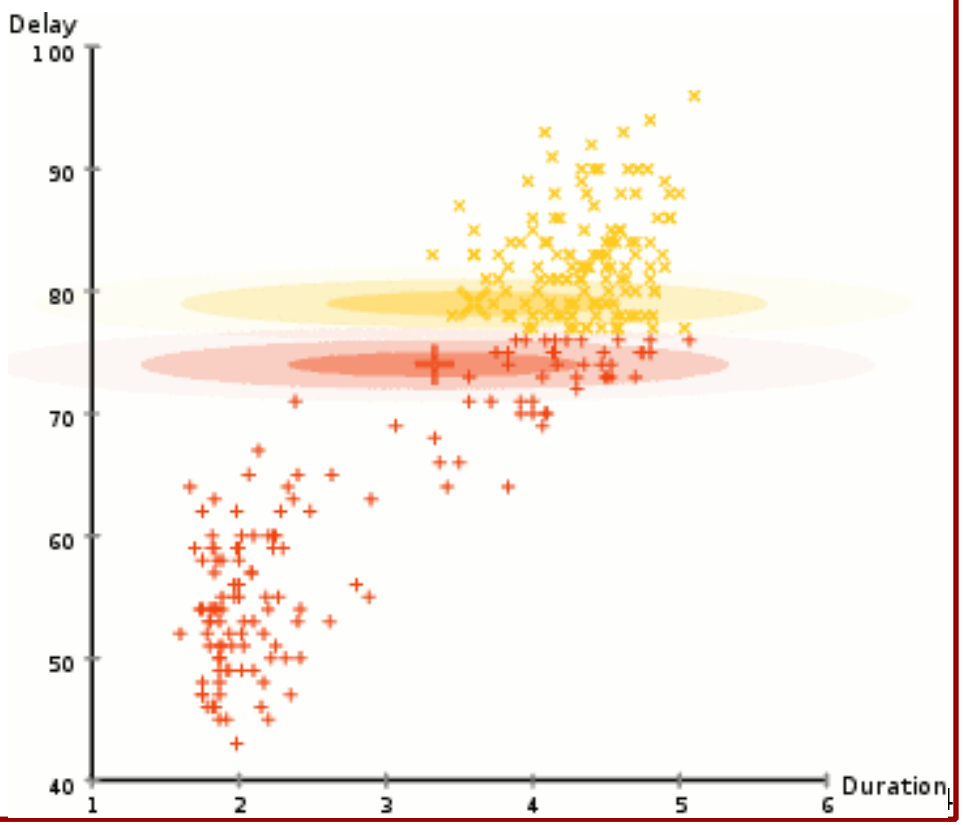
EM for GMM computation

- In short:
  - Initialize GMM parameters (means, variances, weights)
  - *For each data point find the prob it came from each Gaussian*
  - *(In case of hard-GMM membership) assign each datapoint to the most likely Gaussian*
  - *Use the assignment to find the new GMM parameters*
  - Iterate until convergence
  - For GMM ASR features are usually
    - 13 MFCC's
    - First order derivatives
    - Second order derivatives
    - = 39 dimensional features
    - Or use LDA…



EM for GMM computation

- In short:
  - Initialize GMM parameters (means, variances, weights)
  - *For each data point find the prob it came from each Gaussian*
  - *(In case of hard-GMM membership) assign each datapoint to the most likely Gaussian*
  - *Use the assignment to find the new GMM parameters*
  - Iterate until convergence
  - For GMM ASR features are usually
    - 13 MFCC's
    - First order derivatives
    - Second order derivatives
    - = 39 dimensional features
    - Or use LDA…



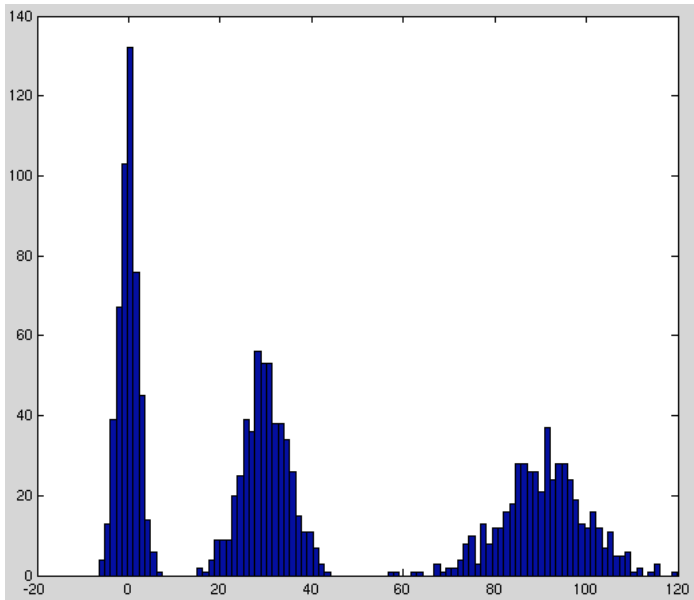
ASR Training: Acoustic Model

- Acoustic model:
  - Represent the variability for each of these 39 numbers for each state
  - Due to multiple sound instantiations/conditions/speakers/... Gaussian is not a good model.
  - Histogram???
  - Preferred method is a Mixture Gaussian model
  - So in summary, typically:
    - Each phoneme is represented by 3 states
    - Each state is represented by 39 dimensions
    - Each dimension is represented by a mixture Gaussian model (N-means, N-variances, and N-mixture weights -- assuming diagonal cov. matrix)

- Complexity of Acoustic model in real numbers:
  - Say 50 phonemes (English)
  - (REAL SYSTEMS) For better accuracy use triphone representation
    - (potentially 50^3 but usually 3-9K triphones)
  - Each of these has 3 states
  - Each of these has 39 representation dimensions
  - Each dimension has about 16 mixture gaussians
  - 5,000*3*39*(16*3) = ~30,000,000 parameters!!

ASR Training: Acoustic Model

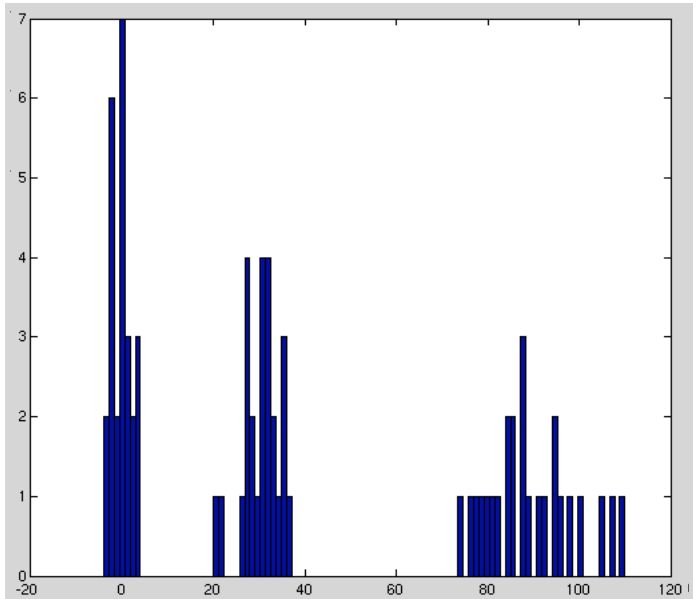
- Acoustic model:
  - Represent the variability for each of these 39 numbers for each state
  - Due to multiple sound instantiations/conditions/speakers/... Gaussian is not a good model.
  - Histogram???
  - Preferred method is a Mixture Gaussian model
  - So in summary, typically:
    - Each phoneme is represented by 3 states
    - Each state is represented by 39 dimensions
    - Each dimension is represented by a mixture Gaussian model (N-means, N-variances, and N-mixture weights -- assuming diagonal cov. matrix)



- Complexity of Acoustic model in real numbers:
  - Say 50 phonemes (English)
  - (REAL SYSTEMS) For better accuracy use triphone representation
    - (potentially 50^3 but usually 3-9K triphones)
  - Each of these has 3 states
  - Each of these has 39 representation dimensions
  - Each dimension has about 16 mixture gaussians
  - 5,000*3*39*(16*3) = ~30,000,000 parameters!!

ASR Training: Acoustic Model

- Acoustic model:
  - Represent the variability for each of these 39 numbers for each state
  - Due to multiple sound instantiations/conditions/speakers/... Gaussian is not a good model.
  - Histogram???
  - Preferred method is a Mixture Gaussian model
  - So in summary, typically:
    - Each phoneme is represented by 3 states
    - Each state is represented by 39 dimensions
    - Each dimension is represented by a mixture Gaussian model (N-means, N-variances, and N-mixture weights -- assuming diagonal cov. matrix)



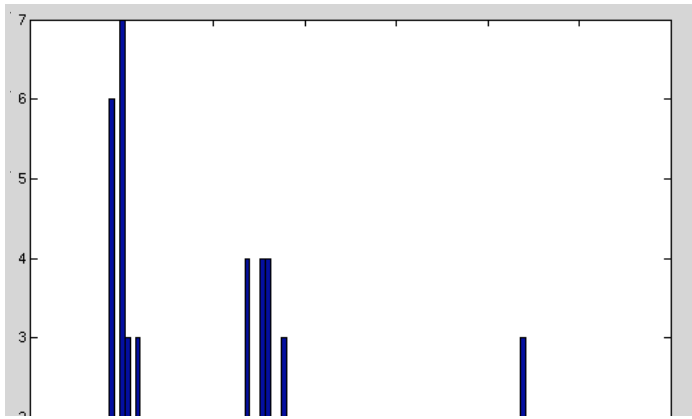
- Complexity of Acoustic model in real numbers:
  - Say 50 phonemes (English)
  - (REAL SYSTEMS) For better accuracy use triphone representation
    - (potentially 50^3 but usually 3-9K triphones)
  - Each of these has 3 states
  - Each of these has 39 representation dimensions
  - Each dimension has about 16 mixture gaussians
  - 5,000*3*39*(16*3) = ~30,000,000 parameters!!

USC Viterbi School of Engineering

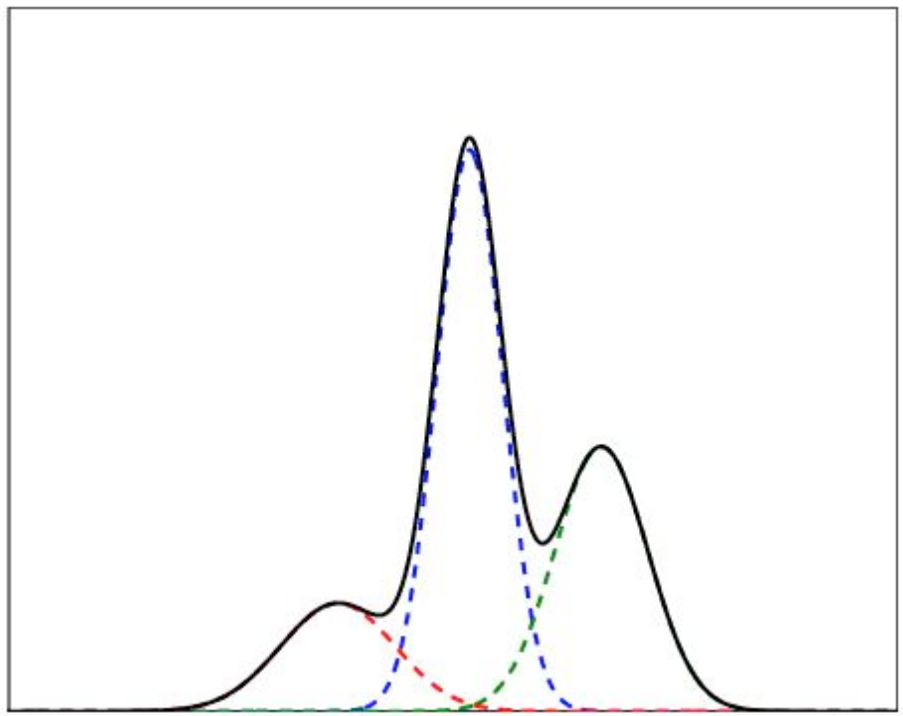
ASR Training: Acoustic Model

SCUBA

- Acoustic model:
  - Represent the variability for each of these 39 numbers for each state
  - Due to multiple sound instantiations/conditions/speakers/... Gaussian is not a good model.
  - Histogram???
  - Preferred method is a Mixture Gaussian model
  - So in summary, typically:
    - Each phoneme is represented by 3 states
    - Each state is represented by 39 dimensions
    - Each dimension is represented by a mixture Gaussian model (N-means, N-variances, and N-mixture weights -- assuming diagonal cov. matrix)



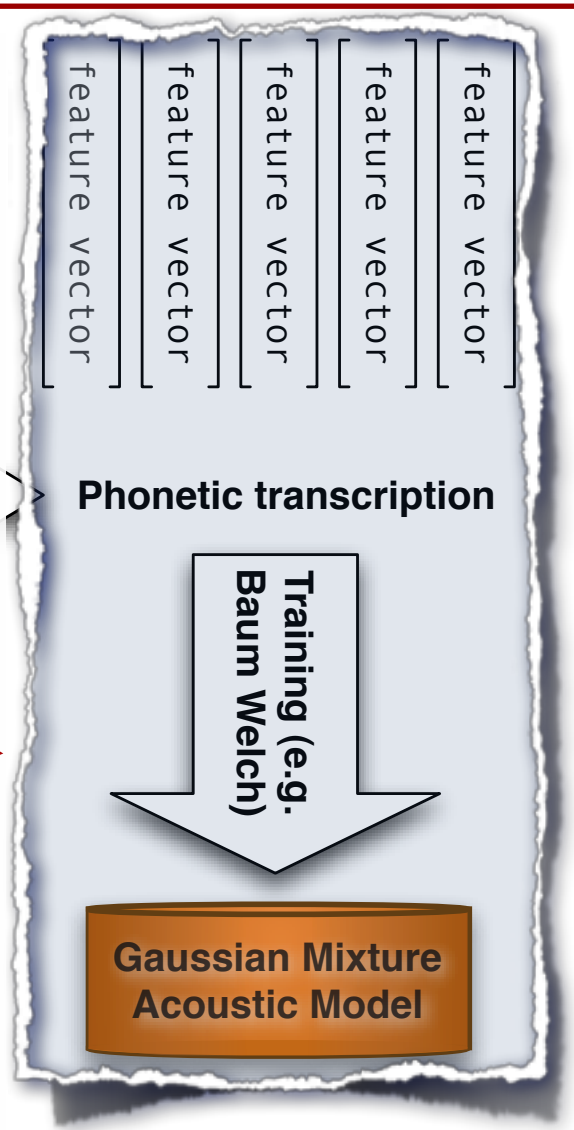
- Complexity of Acoustic model in real numbers:
  - Say 50 phonemes (English)
  - (REAL SYSTEMS) For better accuracy use triphone representation
    - (potentially 50^3 but usually 3-9K triphones)
  - Each of these has 3 states
  - Each of these has 39 representation dimensions
  - Each dimension has about 16 mixture gaussians
  - 5,000*3*39*(16*3) = ~30,000,000 parameters!!



ASR Training: Acoustic Model

- Gives a prior on what speech sounds like
- Needs to match the domain where ASR will be used

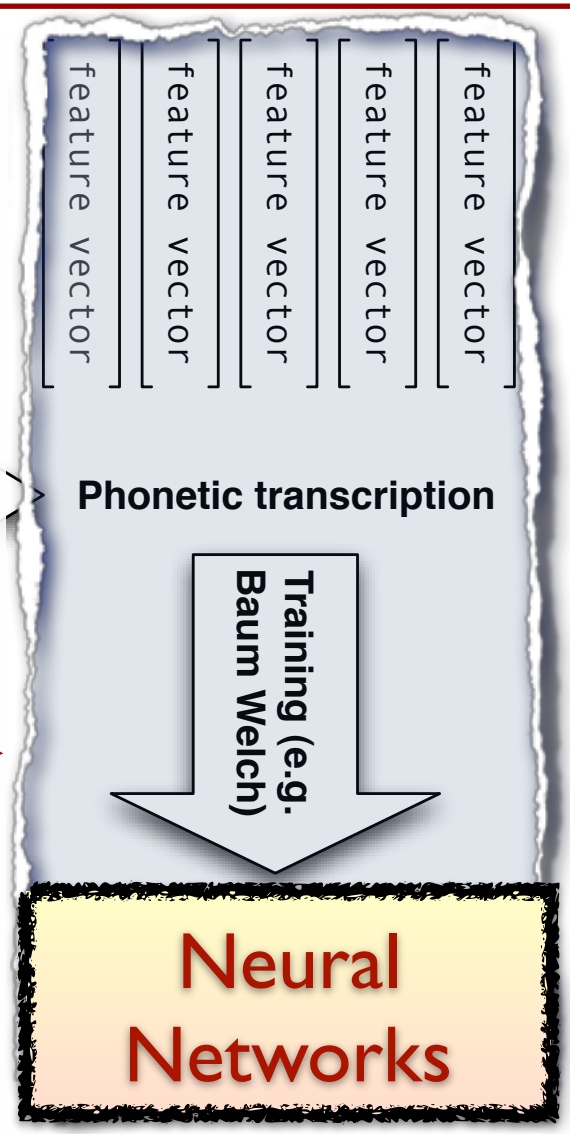
- Usual problem: training data is mismatched with domain data



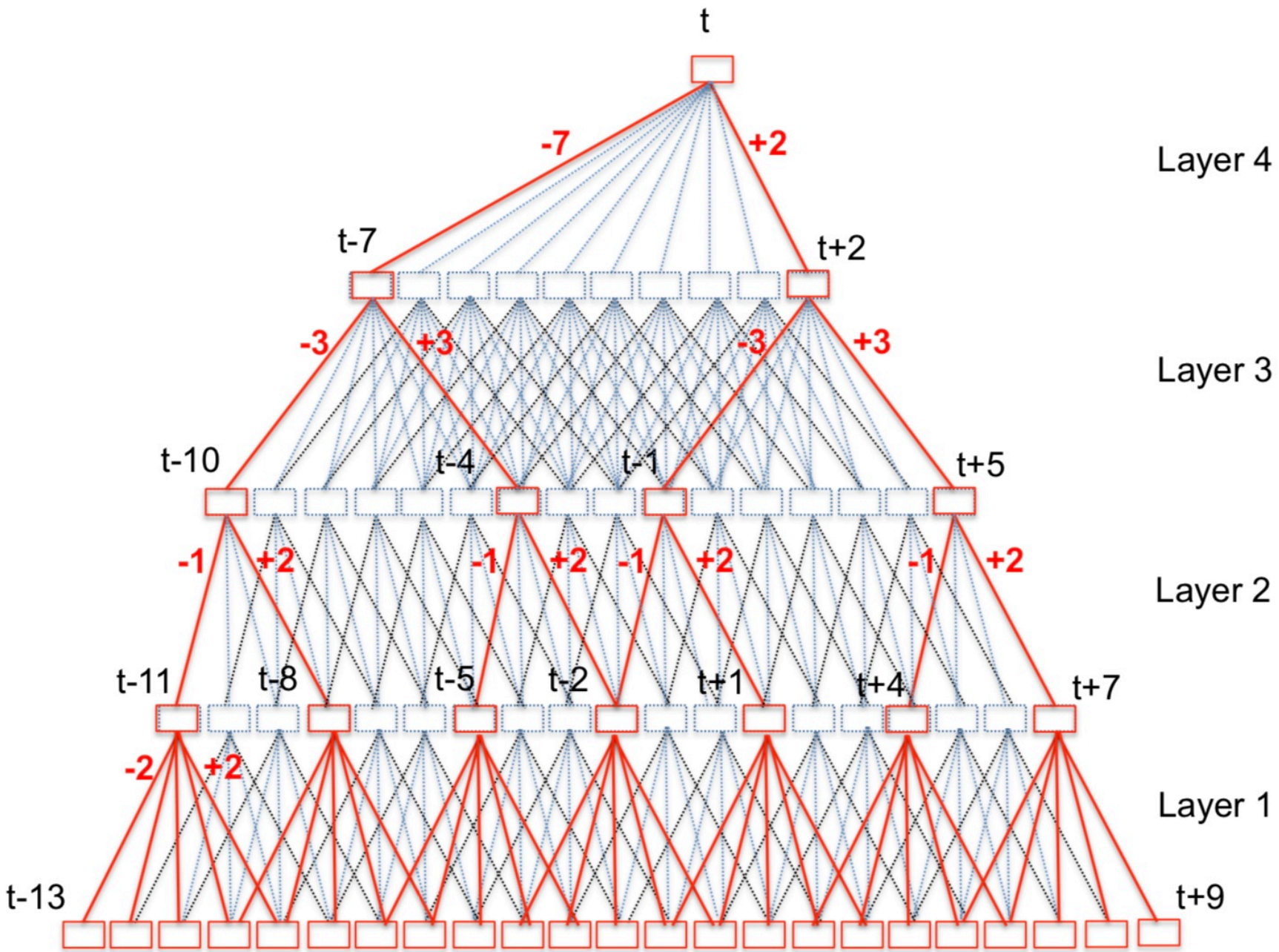
ASR Training: Acoustic Model

- Gives a prior on what speech sounds like
- Needs to match the domain where ASR will be used

- Usual problem: training data is mismatched with domain data

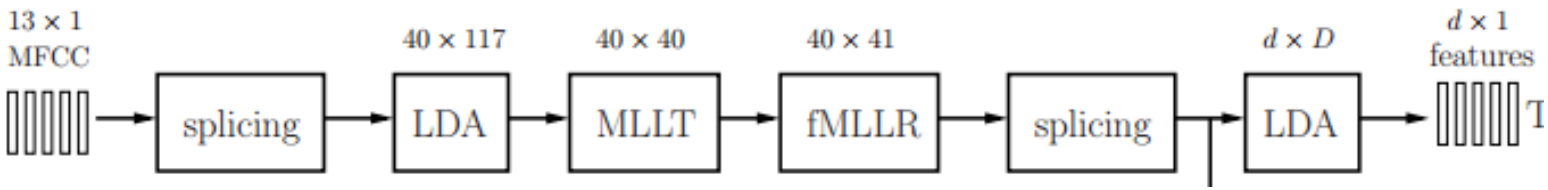


SLIDE TITLE


t
-7
+2
Layer 4
t-7
t+2
-3
+3
-3
+3
Layer 3
t-10
t-4
t-1
t+5
-1
+2
-1
+2
-1
+2
-1
+2
Layer 2
t-11
t-8
t-5
t-2
t+1
t+4
t+7
-2
+2
Layer 1
t-13
t+9

KALDI – DNN AM

- **Input Features / Dimension:**



- **P-norm Non-linearity:**
  - Are an extension of maxout networks.
  - Maxout network: $y = \max_{i=1}^{G} x_i$
  - Soft-maxout: $y = \log \sum_{i=1}^{G} \exp(x_i)$
  - p-norm: $y = \|\mathbf{x}\|_p = \left(\sum_i |x_i|^p \right)^{1/p}.$

- **Normalization Layers:**

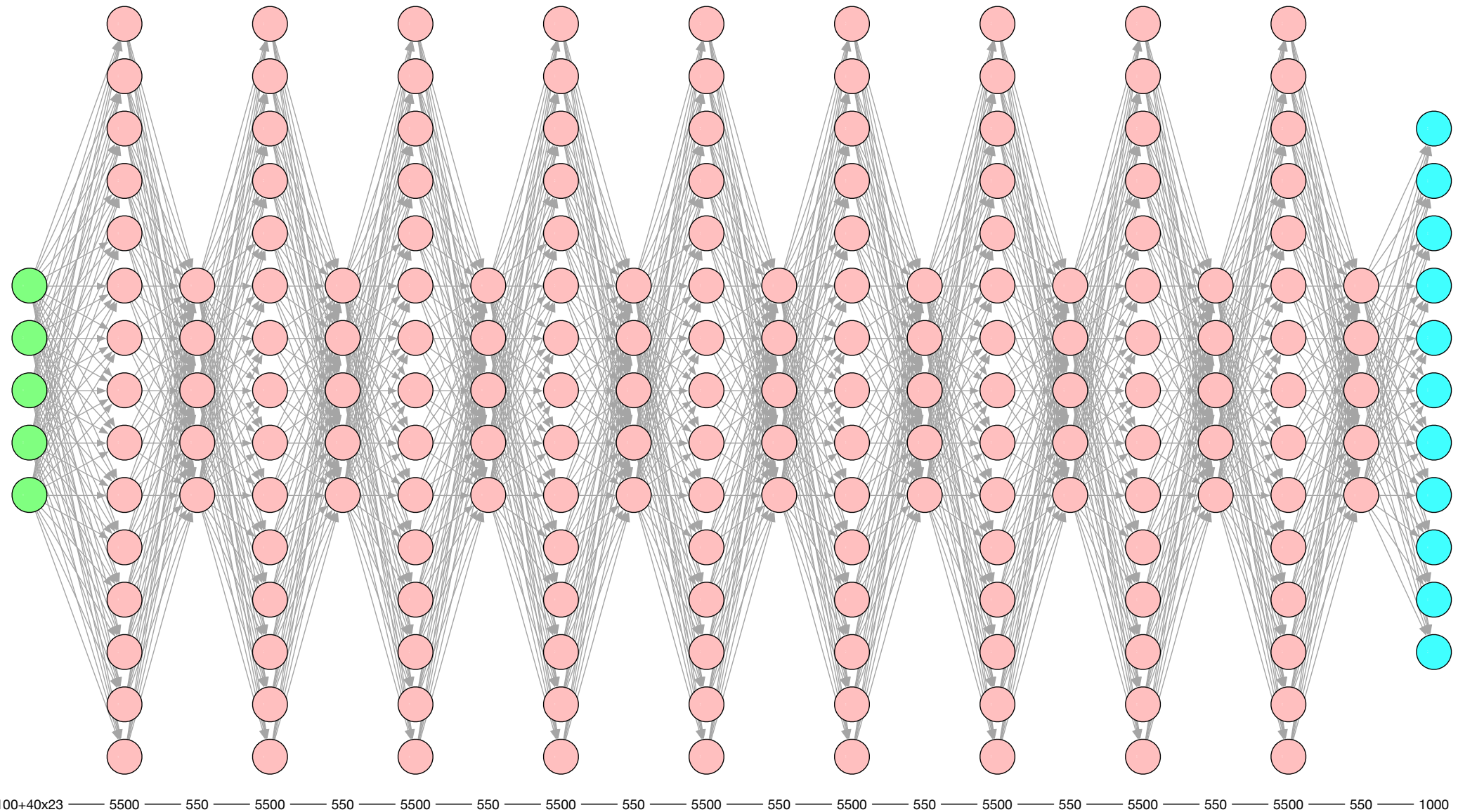
$$\boldsymbol{y_i} = \begin{cases} \boldsymbol{x_i}, & \sigma \leq \mathbf{1} \\ \boldsymbol{x_i}/\sigma, & \sigma > \mathbf{1} \end{cases}$$

σ is the uncentered standard deviation of input x_i.

USC Viterbi School of Engineering

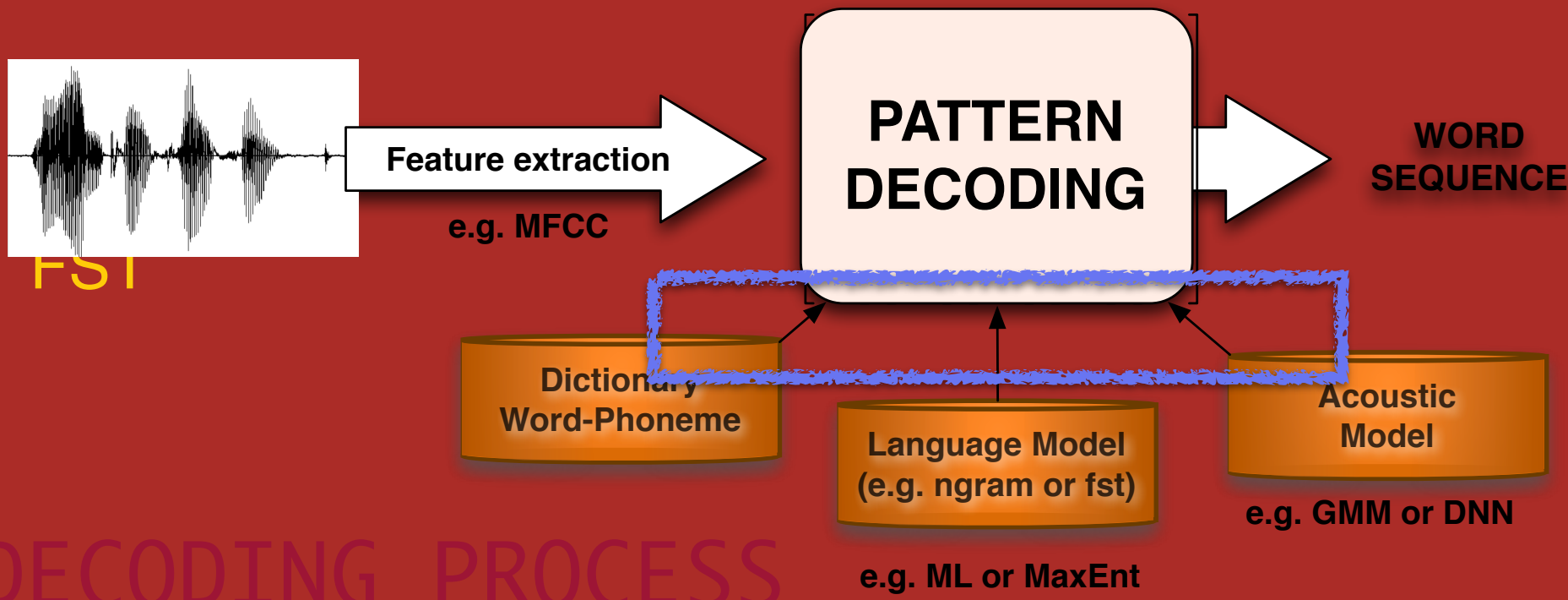
ASR Training: Acoustic Model

SCUBA



HMM 101

- FST



DECODING PROCESS

SPEECH RECOGNITION/HMM101

- "Transitioning" to Neural Nets
- Many things are still not working as great with Neural Nets in speech
- A lot of structure and human knowledge that can be introduced in task
- Machine learning is usually almost 100% data driven
- Language has phonemes, dictionaries, words, sentences, speakers, accents, …
- All these better still with traditional tools
- Opportunities for research
- But also a very busy and fast changing field
-

It's Los Angeles:

Most likely

$P(\text{Rainy Day}) \simeq 0.05$

(or lower!)

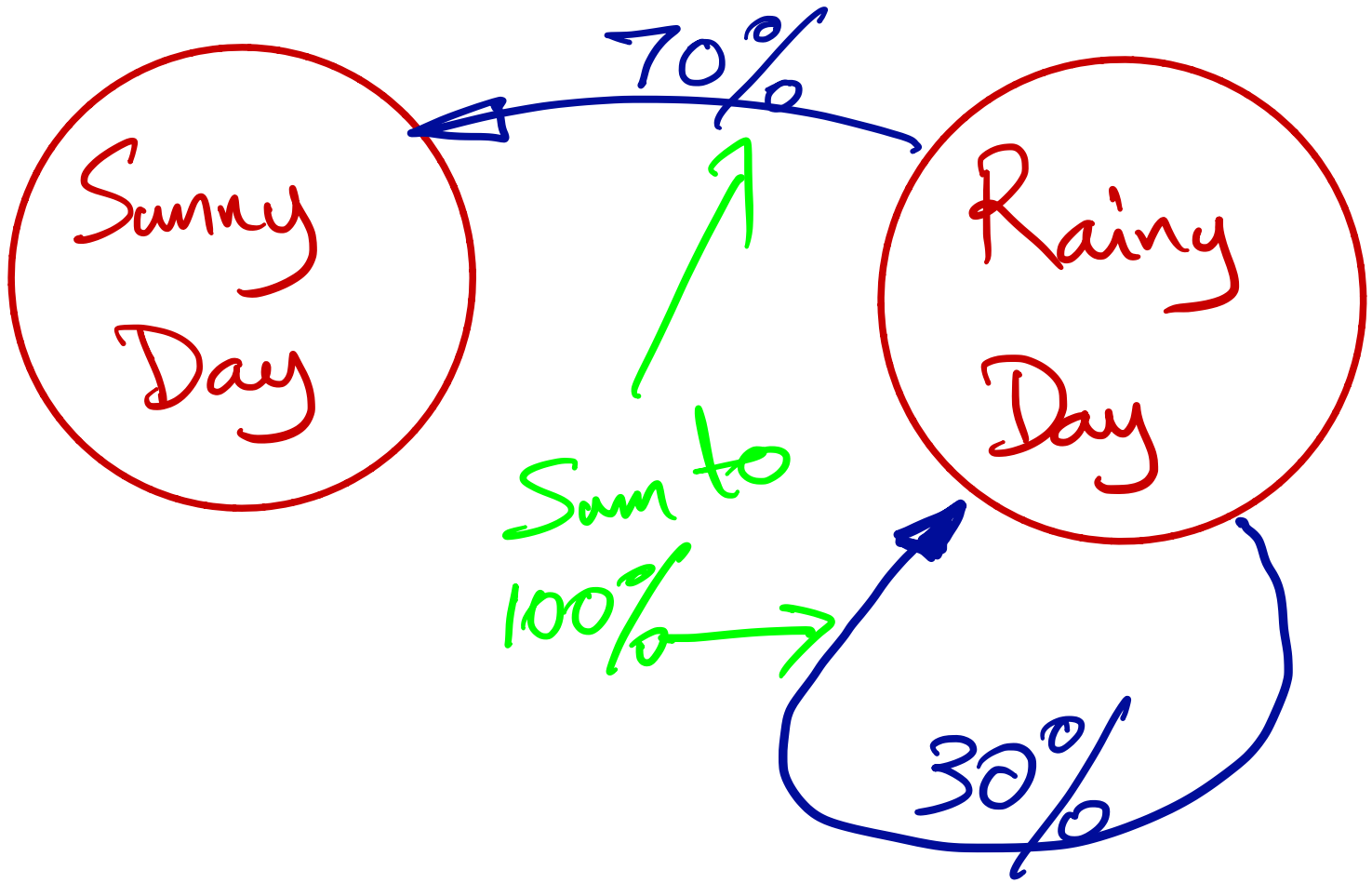
But it's not always sunny in L.A.

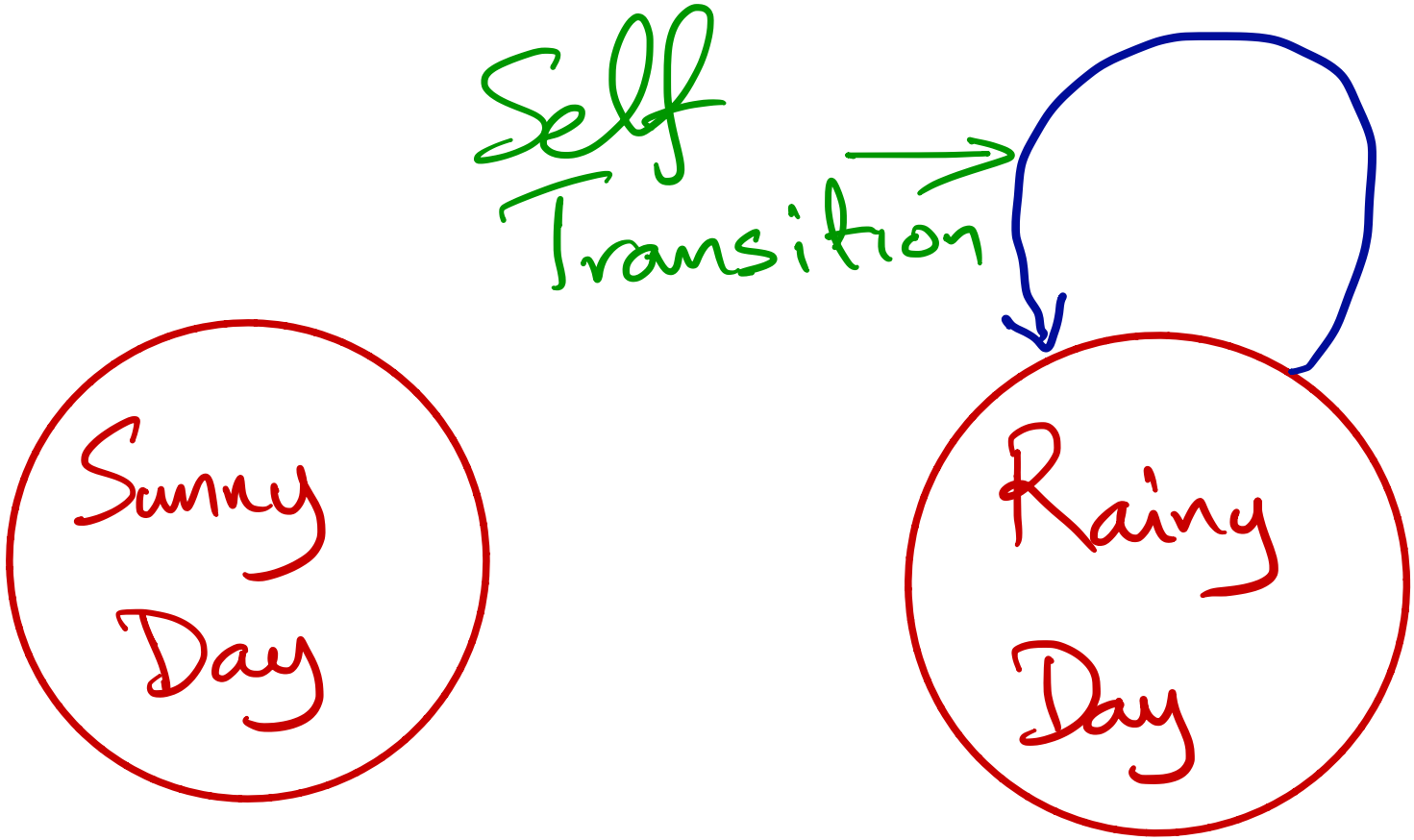
What if yesterday it rained?

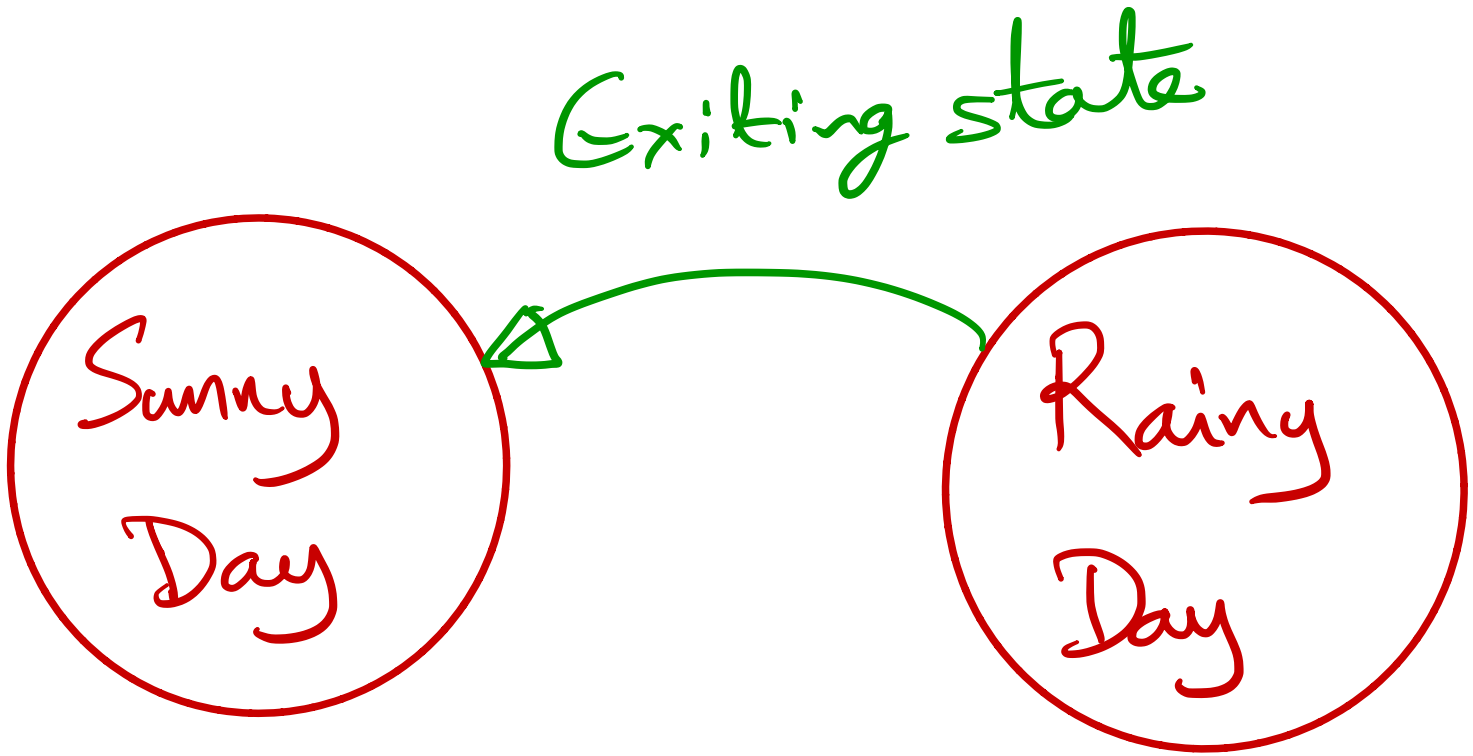
Chances are closer to 50/50.

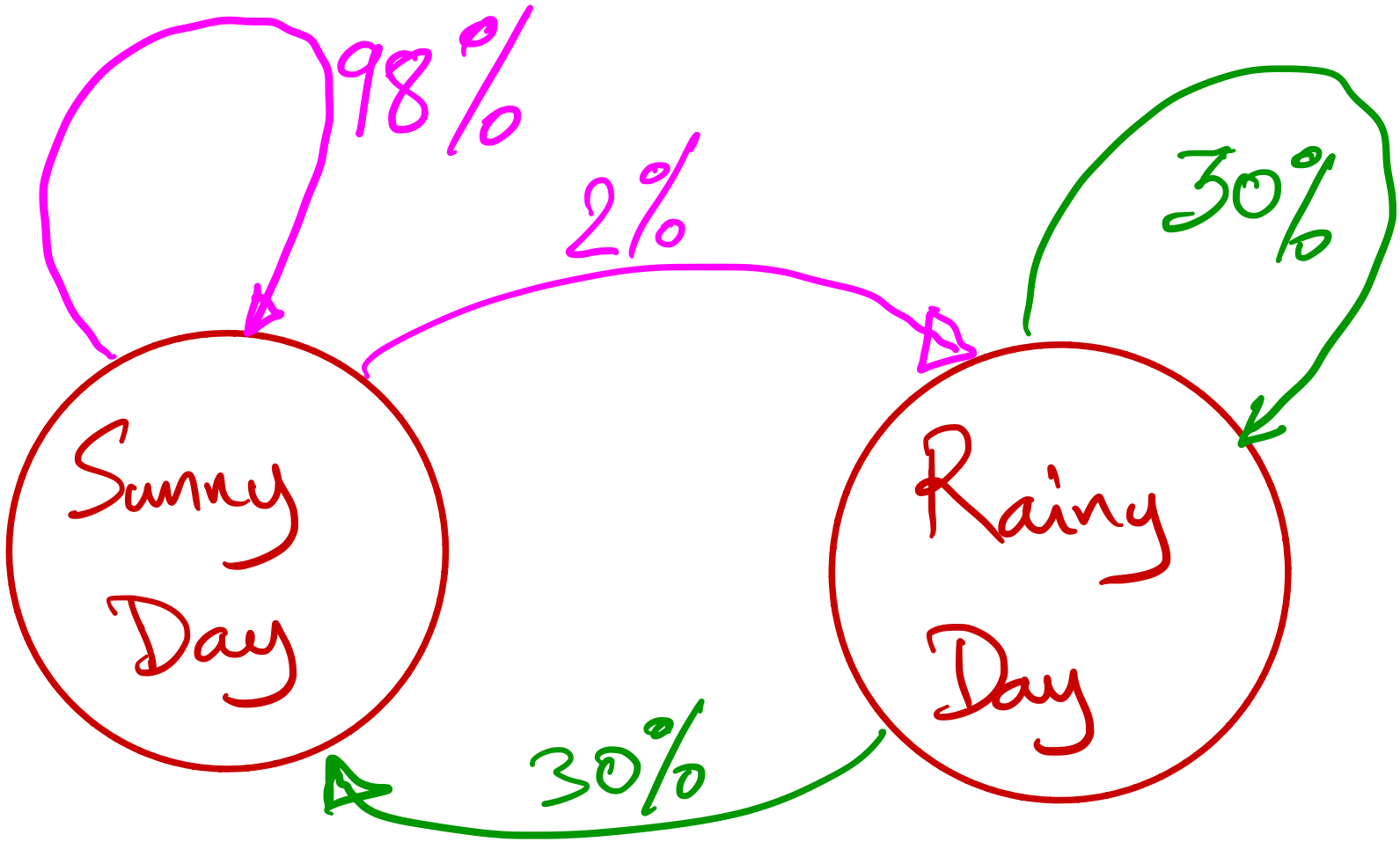
→ say 30/70.

Rainy days are likely to follow rainy days with Prob 0.5.



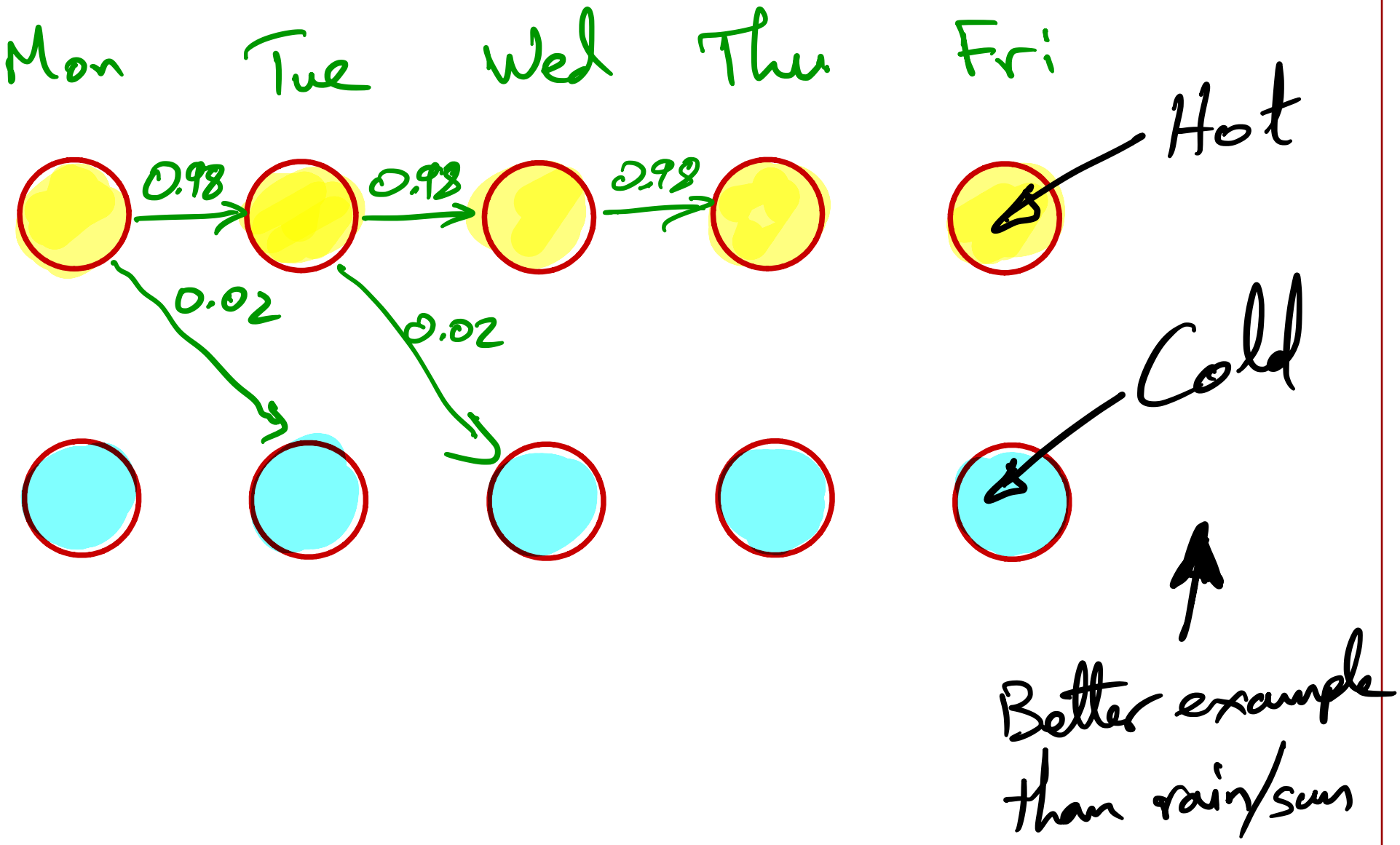

Self
Transition
Sunny
Day
Rainy
Day


Exiting state
Sunny Day
Rainy Day



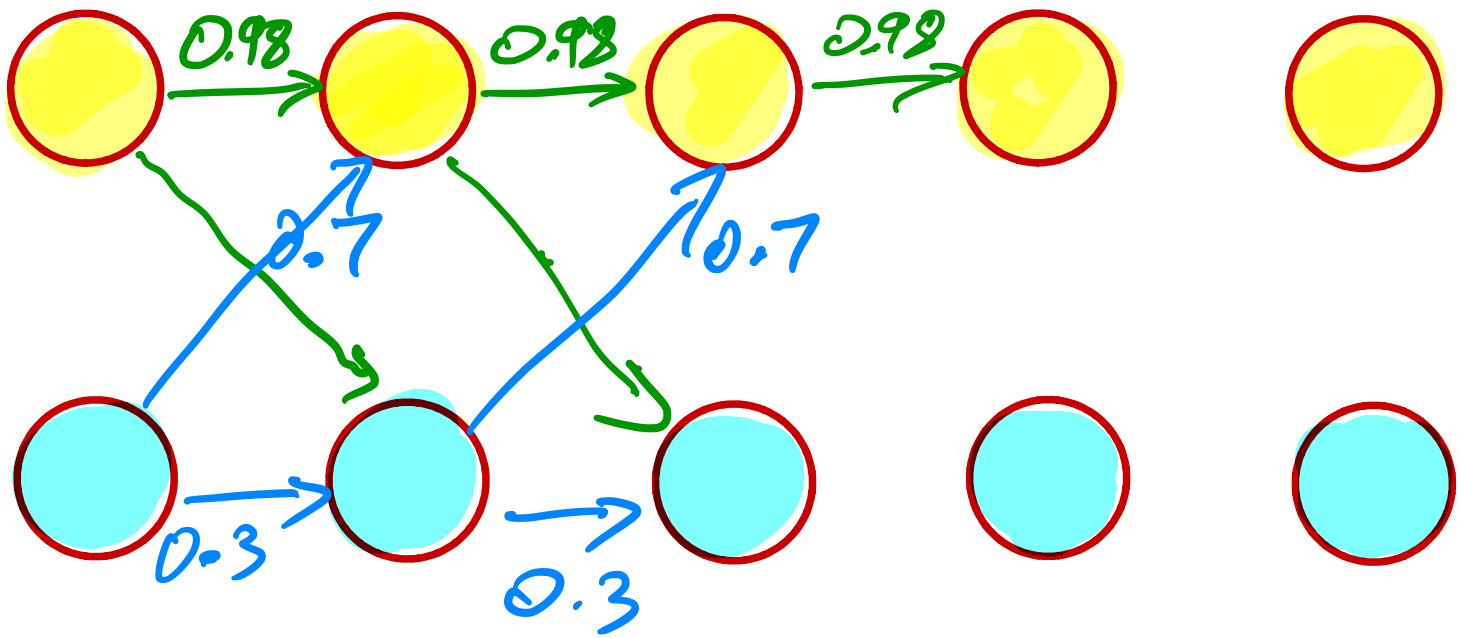
Note that system has memory of 1 Day. → 1st Order Markov Model

Lets unroll this in time (days)



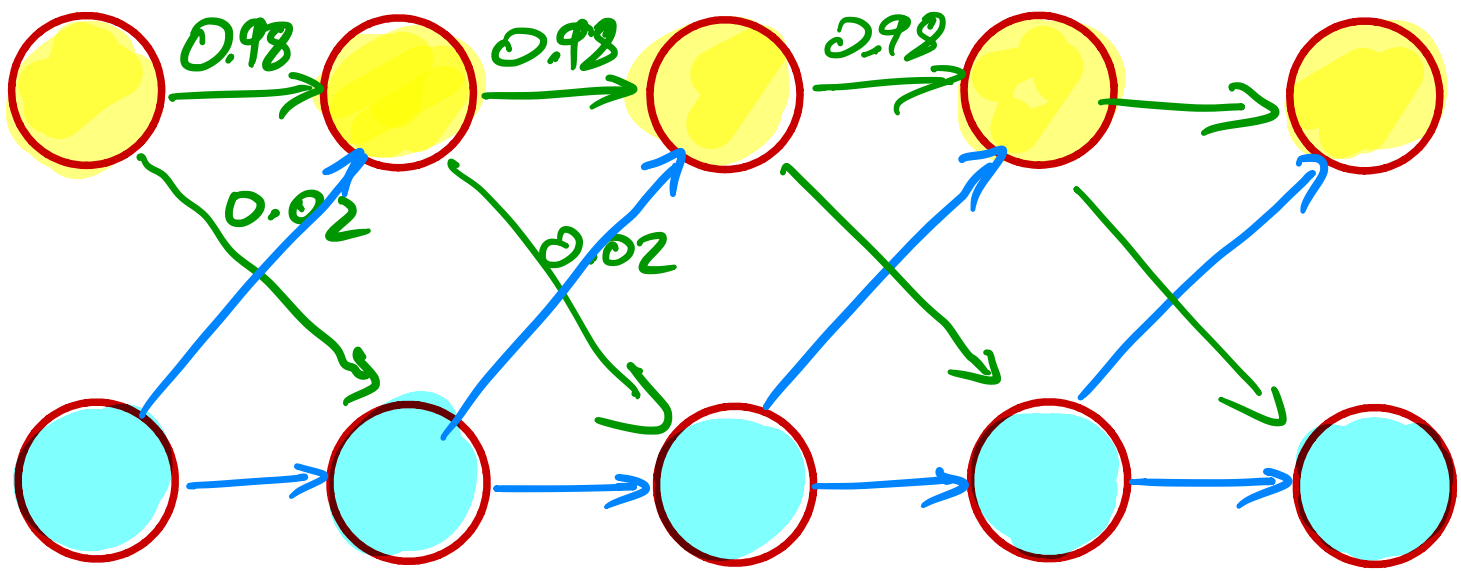
Lets unroll this in time (days)

Mon Tue Wed Thu Fri



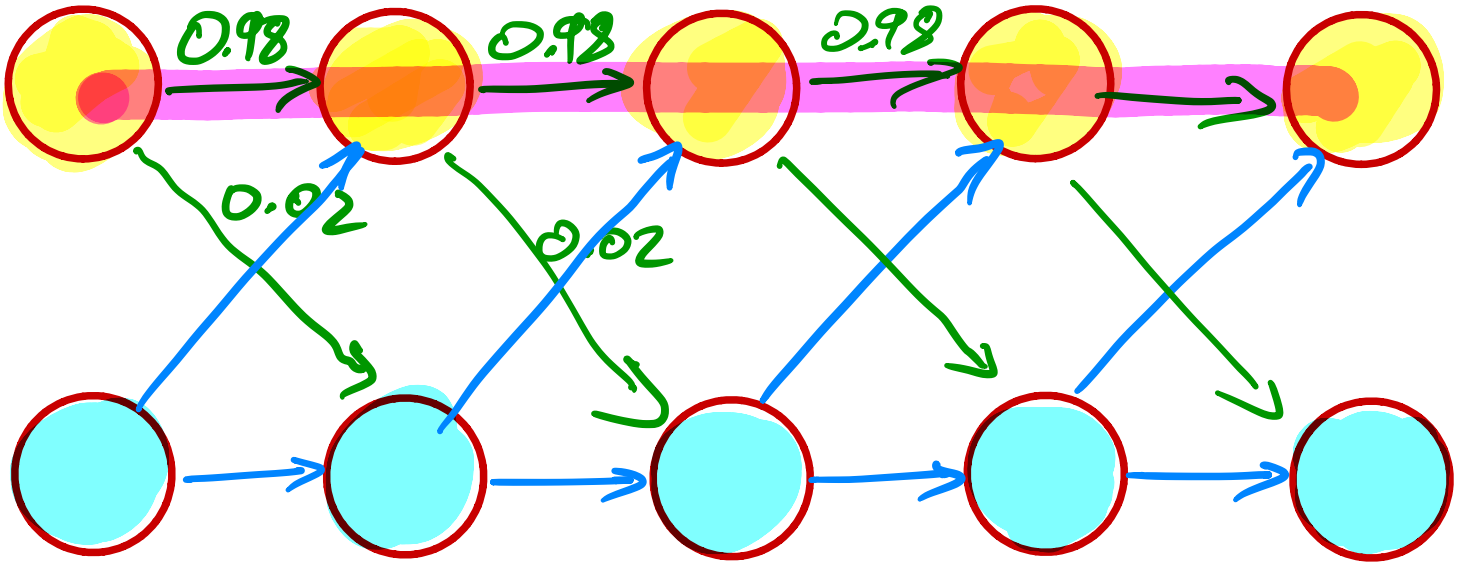
Lets unroll this in time (days)

Mon Tue Wed Thu Fri



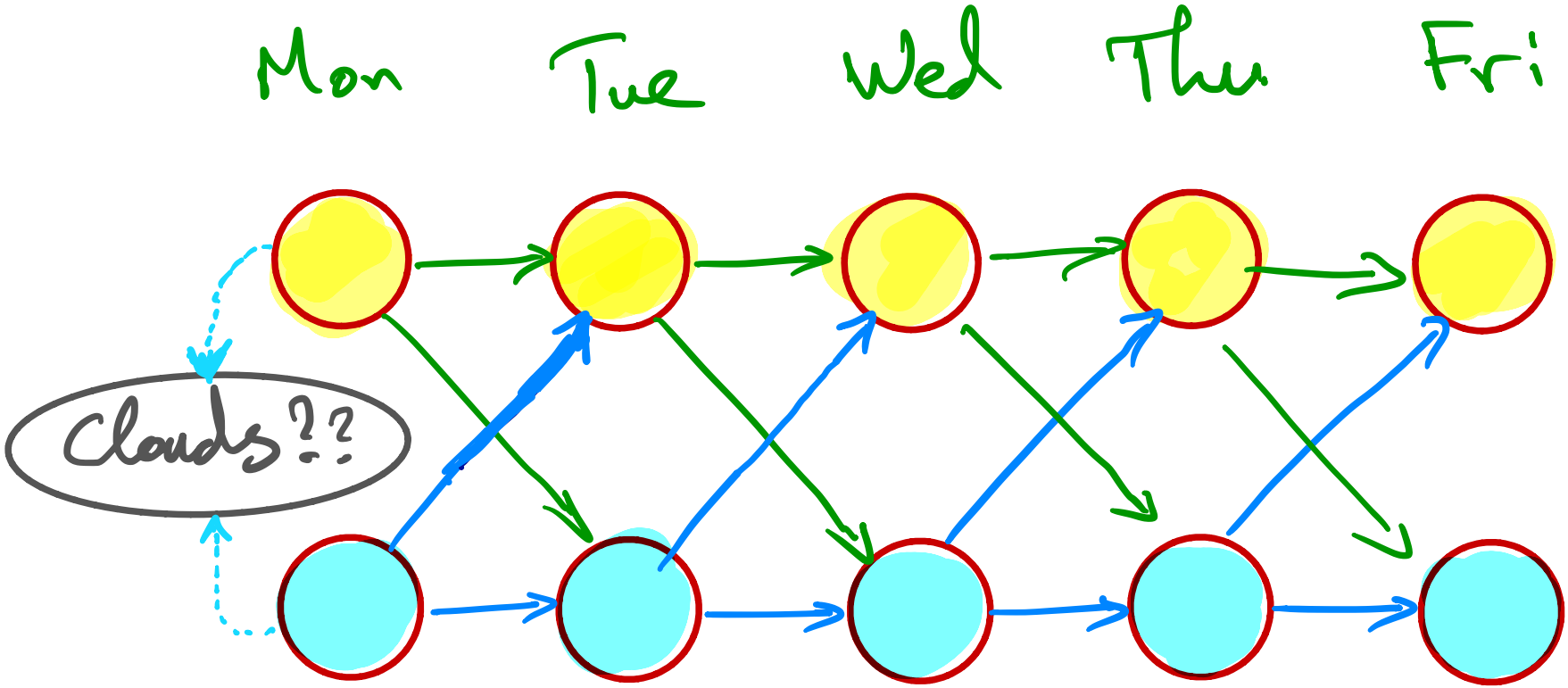
Lets unroll this in time (days)

Mon Tue Wed Thu Fri



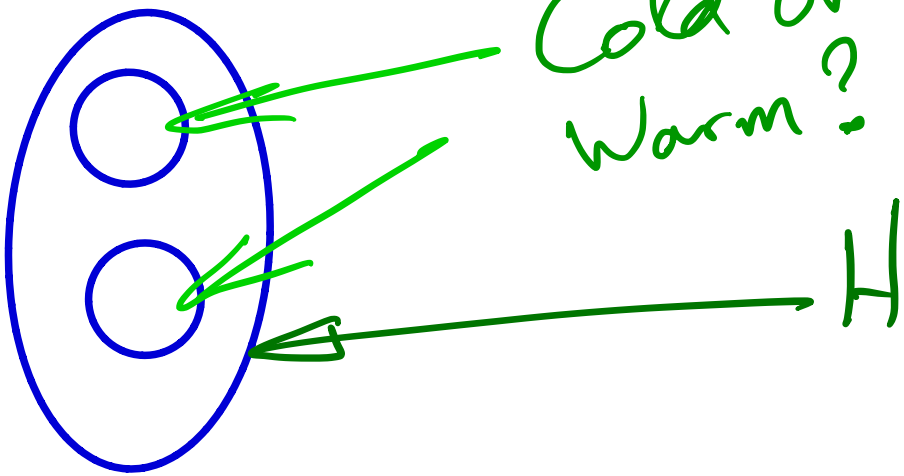
Best Path

But not always !



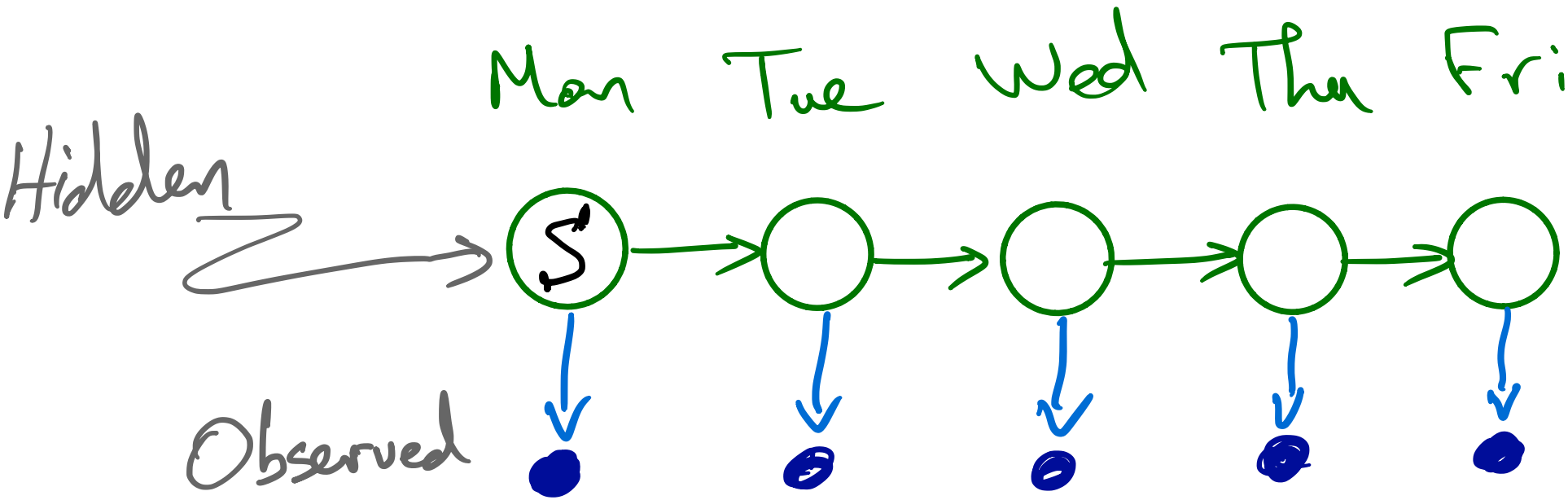
In real world, we would judge based on many observations.

E.g. If its cloudy its likely to be "cold"



Changed from Rain/Sun to something Hidden if I am inside.

Observations: eg. sunny, cloudy, rain, trees moving from wind, ...

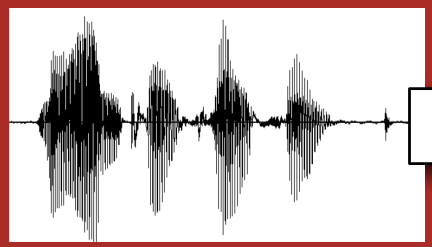


Usual unroled view of an HMM

HMMs can be represented with FST

..... so lets get to that.

Language Models (arpa/srilm)



Feature extraction

e.g. MFCC

PATTERN DECODING

WORD SEQUENCE

Dictionary Word-Phoneme

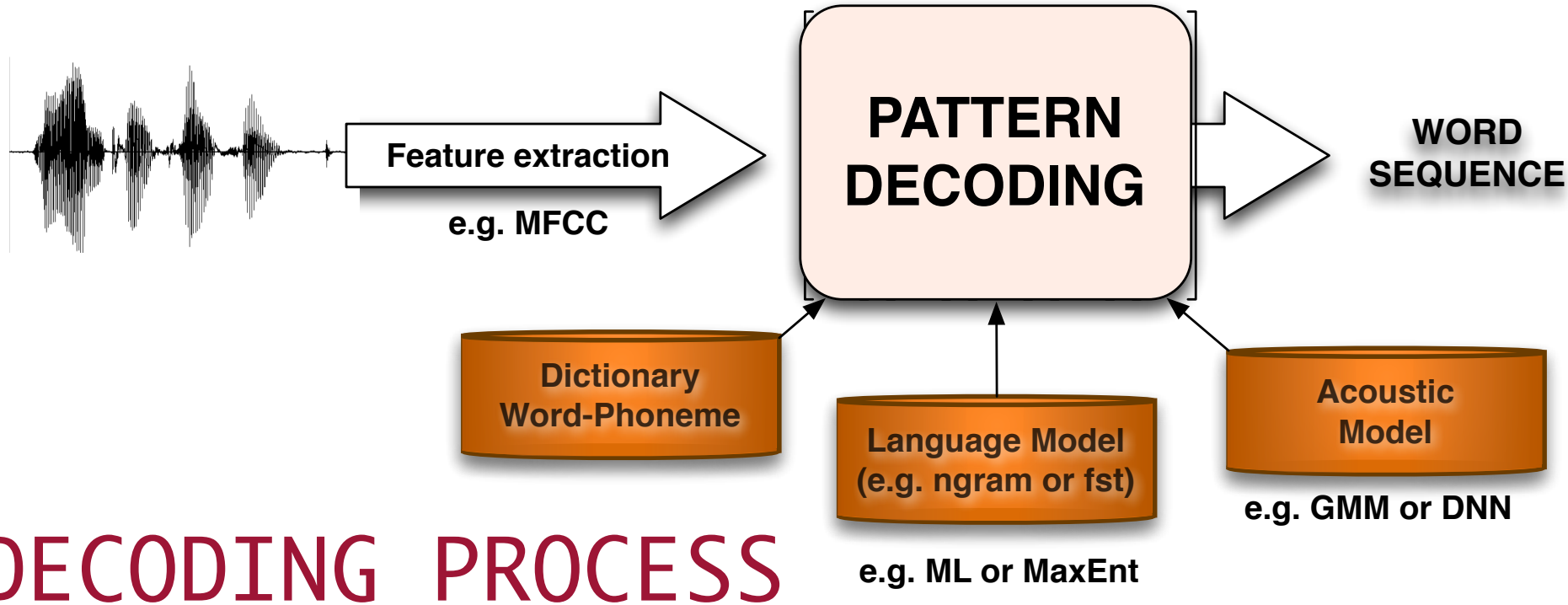
Language Model (e.g. ngram or fst)

e.g. ML or MaxEnt

Acoustic Model

e.g. GMM or DNN

DECODING PROCESS



DECODING PROCESS

Language Model

- P(W) can be extracted from existing text:
- For simplicity and feasibility approximate with:

$$P(W) = P(W_1, W_2, ...W_n) = P(W_1)P(W_2|W_1)P(W_3|W_1W_2....)P(W_n|W_1W_2...W_{n-1})$$

$$P(W) = P(W_1, W_2, ...W_n) = P(W_1)....P(W_{n-1}|W_{n-3}W_{n-2})P(W_n|W_{n-2}W_{n-1})$$

- When we don't have enough data - next best:

$$p(w_3|w_1, w_2) = \begin{array}{rl} \text{if(trigram exists)} & P_3(w_1, w_2, w_3) \\ \text{else if(bigram } w_1, w_2 \text{ exists)} & BOW(w_1, w_2)P(w_3|w_2) \\ \text{else} & P(w_3|w_2) \end{array}$$

SLIDE TITLE

- P(w2|w1)=p(w1,w2)/P(w1)

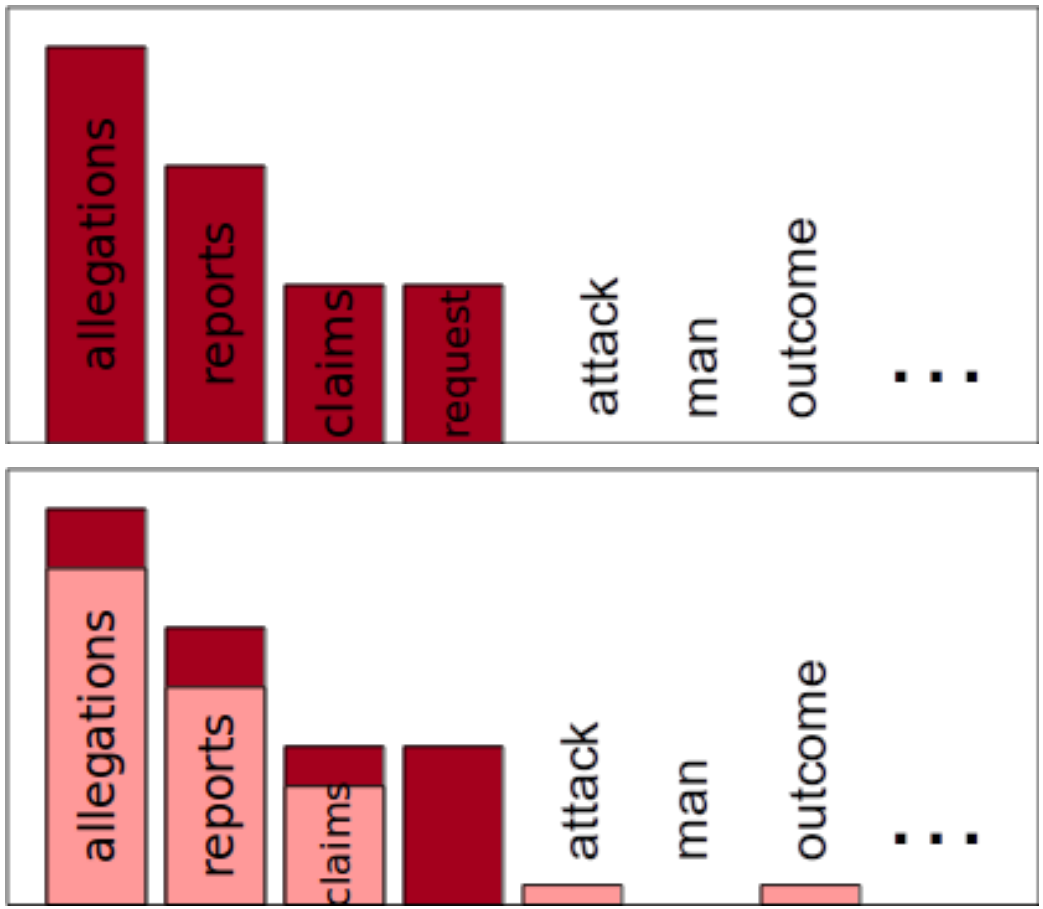
- Example
- P(And|<s>)=3/8
- P(<s>|spider)=1/3

<s> The itsy-bitsy spider </s>
<s> Climbed up the water spout </s>
<s> Down came the rain </s>
<s> And washed the spider out </s>
<s> Out came the sun </s>
<s> And dried up all the rain </s>
<s> And the itsy-bitsy spider </s>
<s> Climbed up the spout again </s>

- This is problematic for higher order ngrams
- Many will never be seen.
- Even here P(<s>|dried)=0. That means “<s> The spout dried </s>” has prob=0
- Need smoothing, interpolation, and backoff.

SMOOTHING

- When we have sparse statistics
- P(w | denied the)
  - 3 allegations
  - 2 reports
  - 1 claims
  - 1 request
- Steal probability mass
  - 2.5 allegations
  - 1.5 reports
  - 0.5 claims
  - 0.5 request
  - 2 other



- Many smoothing methods: Add-One smoothing, Add-K smoothing, Unigram Prior smoothing, Good-Turing smoothing, Kneser-Ney smoothing, Witten-Bell
  - P(w2|w1)=(p(w1,w2)+*something*) / (P(w1)+*something else*)

- Terms:
  - unigram
  - bigram (1st order markov model)
  - trigram (2nd order markov model)
  - . . .
- log likelihood: how likely is a sequence
- perplexity: how (un)likely is the sequence, normalized by the number of words
  - ppl(n words sequence)=1/(prob n words sequence)^1/n
  - In log:
  - ppl(w) = -Sum(log(prob) of all the ngrams)/n
  - Better to compare un-even length inputs.
  - `$ echo "i am very sick today " | ngram -lm /auto/rcf-42/georgiou/hpcc599/georgiou/egs/tedlium/s5/db/cantab-TEDLIUM/cantab-TEDLIUM-pruned.lm3.gz -ppl -`
  - `file -: 1 sentences, 5 words, 0 OOVs`
  - `0 zeroprobs, logprob= -11.1839 ppl= 73.11111 ppl1= 172.4964`
  - `$ echo "this is a very long sentense </s> <s> i am very ill today how about you are you feeling well" | ngram -lm /auto/rcf-42/georgiou/hpcc599/georgiou/egs/tedlium/s5/db/cantab-TEDLIUM/ca`
  - `file -: 1 sentences, 20 words, 0 OOVs`
  - `0 zeroprobs, logprob= -45.3783 ppl= 144.8343 ppl1= 185.7441`
  - `georgiou@hpc3655:~/hpcc599/georgiou/fst-kaldi$`
  -

SRILM 101

- . path.sh

- echo """The itsy-bitsy spider
- Climbed up the spout again""" | tr [a-z] [A-Z] | sed "s/-/ /g"> text_in.txt

- ngram-count -order 1 -text text_in.txt -lm lm.1.arpa
- ngram-count -order 2 -text text_in.txt -lm lm.2.arpa
- ngram-count -order 3 -text text_in.txt -lm lm.3.arpa

- echo """Hello darkness, my old friend
- Disturb the sound of silence""" | tr [a-z] [A-Z] | sed "s/-/ /g;s/,/ /g"> text_in2.txt

- ngram-count -order 3 -text text_in2.txt -lm lm2.3.arpa
- less lm2.3.arpa

- ngram -lm lm.3.arpa -mix-lm lm2.3.arpa -write-lm lm_both.3.arpa

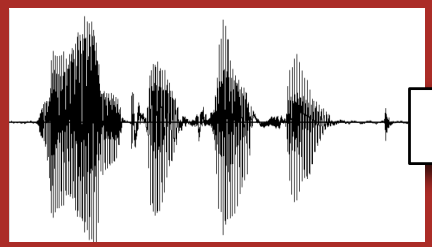
- arpa2fst lm_both.3.arpa > G.fst
- fstdraw -portrait G.fst | dot -Tpdf > G.pdf

INTERPOLATION FROM MULT. DOMAINS

- ngram -lm lm.3.arpa -mix-lm lm2.3.arpa -write-lm lm_both.3.arpa
- This interpolation a good trick to use more data in a domain with little data
- Use a heldout and test set to chose mixing weights.

- -lm: file in ARPA LM format
- -mix-lm: LM to mix in
- -lambda: mixture weight for **-lm**
- Default value: 0.5
- -mix-lm2: second LM to mix in
- -mix-lambda2: mixture weight for -mix-lm2
- Default value: 0
- -mix-lm3: third LM to mix in
- -mix-lambda3: mixture weight for -mix-lm3
- Default value: 0
- . . .
-

Language Modeling with Neural Networks



Feature extraction

e.g. MFCC

PATTERN DECODING

WORD SEQUENCE

Dictionary
Word-Phoneme

Language Model
(e.g. ngram or fst)

e.g. ML or MaxEnt

Acoustic Model

e.g. GMM or DNN

DECODING PROCESS

Outline

1) Review previous material
   - Language Modeling
   - n-gram Language Models

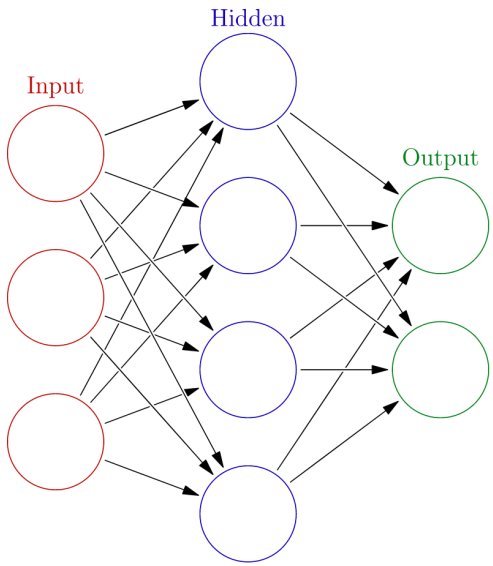
2) Neural Networks
   - Feedforward, Recurrent, Convolutional Language Models
   - Recurrent Neural Network, Long Short-Term Memory

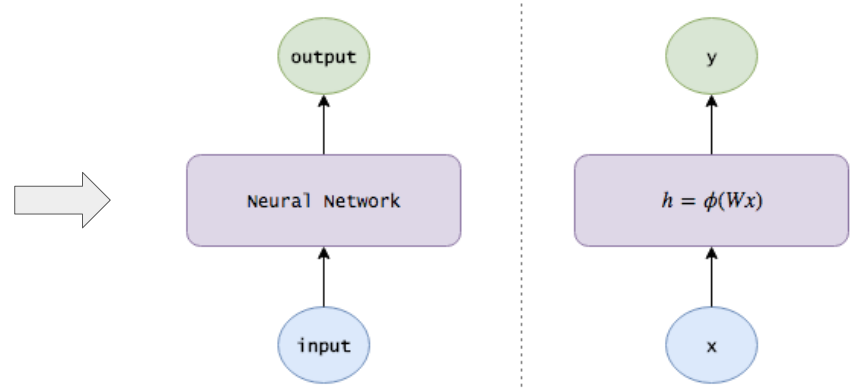
3) Long Short-Term Memory Recurrent Neural Network Language Models
   - Train, generate text
   - Compare against n-gram Language Models
   - General comparison: n-gram vs Feedforward vs Recurrent vs Convolutional

Neural Networks

Neural Network (NN)

- Connectionist model that uses mathematical function to map input to output





Input: x
Output: y
Hidden: h
Parameters: W,V
Activation: ϕ

$$h = \phi(Wx)$$
$$y = Vh$$

- Versatile, scalable & efficient training, work well
- Very popular nowadays

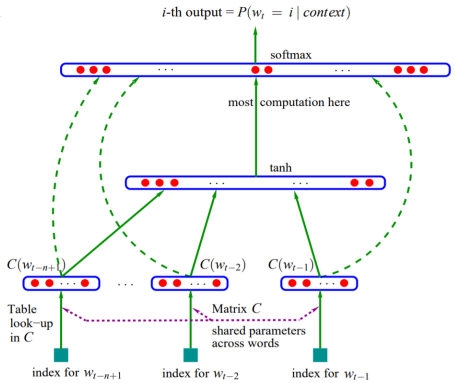
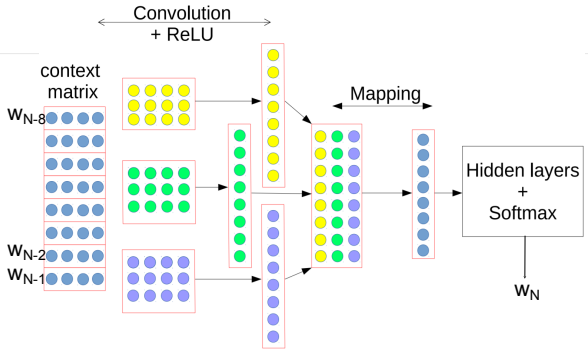
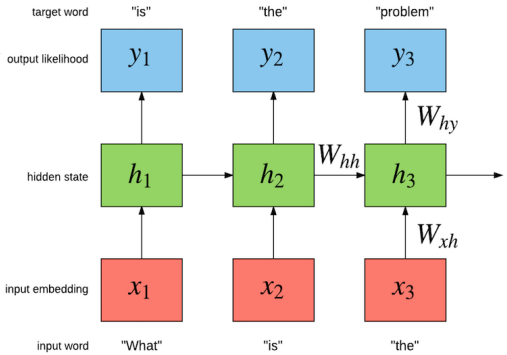
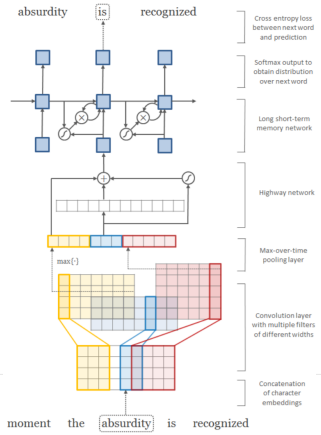
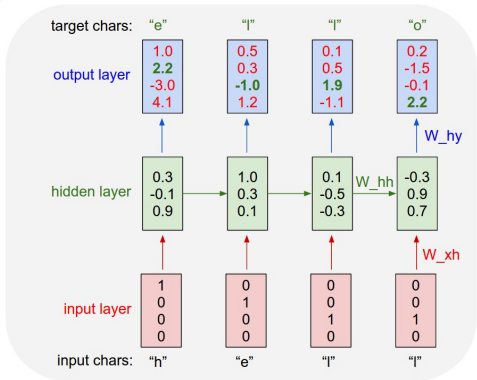
Wikipedia "Artificial neural network" https://en.wikipedia.org/wiki/Artificial_neural_network
Edwin Chen "Exploring LSTMs" http://blog.echen.me/2017/05/30/exploring-lstms/

Neural Network Language Models (NN-LMs)

- Learn complicated function mapping previous words to current word
- Operate in continuous space
  - Notion of similarity for words *"bedroom"*, *"kitchen"* ⇒ can better handle OOV words
- Model size does not increase exponentially with length of context
  - Linear for Feedforward, Convolutional; Constant for Recurrent
- Can better learn long-term dependencies
  - "The computer which I had just put into the machine room on the fifth floor crashed"

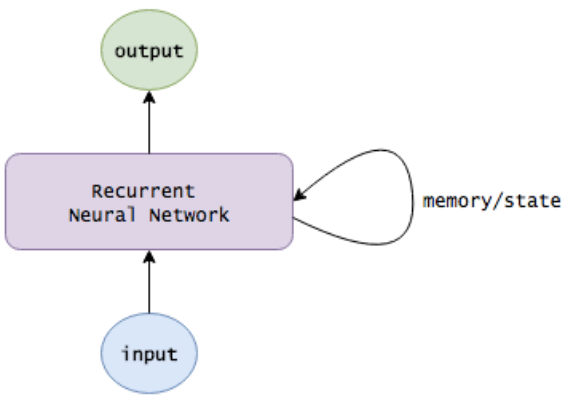
Bengio et al. "A Neural Probabilistic Language Model" http://www.jmlr.org/papers/volume3/bengio03a/bengio03a.pdf
Abigail See "Recurrent Neural Networks and Language Models" https://web.stanford.edu/class/cs224n/lectures/lecture8.pdf

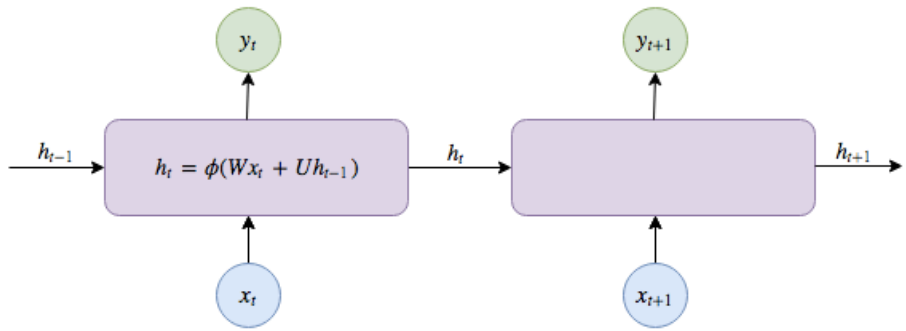
Different classes of NN-LMs

	Feedforward	Convolutional	Recurrent
Word level	 **Bengio et al.** "A Neural Probabilistic Language Model" http://www.jmlr.org/papers/volume3/bengio03a/bengio03a.pdf	 **Pham et al.** "Convolutional Neural Network Language Models" https://www.aclweb.org/anthology/D16-1123	 **Nicholas Leonard** "Language modeling a billion words" http://torch.ch/blog/2016/07/25/nce.html
Character level	 **Kim et al.** "Character-Aware Neural Language Models" https://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/download/12489/12017		 **Andrej Karpathy** "The Unreasonable Effectiveness of Recurrent Neural Networks" http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Recurrent Neural Network (RNN)

- Neural Network with memory





Input: x_t
Output: y_t
Hidden: h_t
Parameters: W,V,U
Activation: ϕ

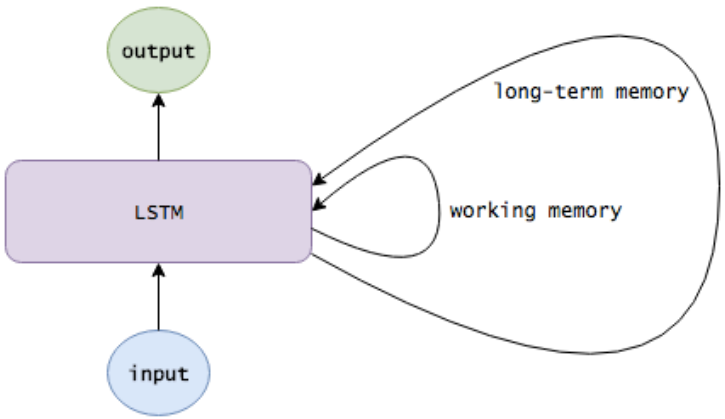
- "Hidden State" h_t stores information from past
- Used for modeling time series, sequences
- In theory: well-suited for capturing long-term dependencies
  In practice: difficult to train because of vanishing/exploding gradient problem

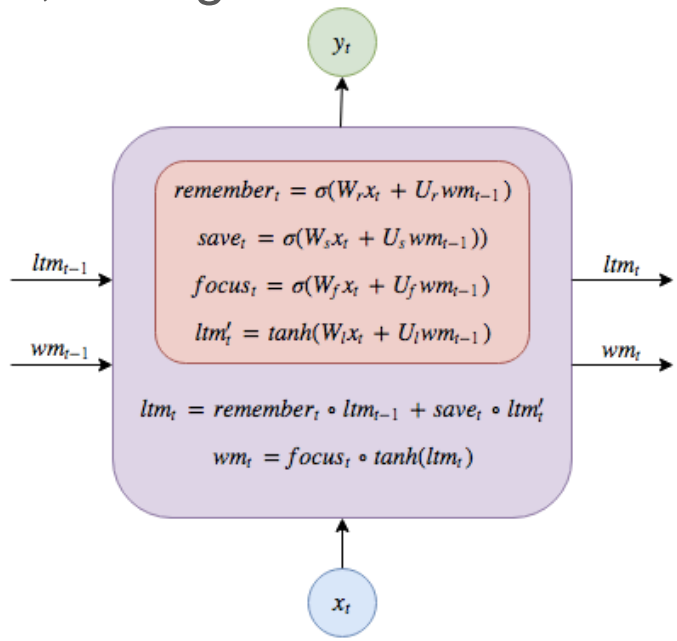
$$h_t = \phi(Wx_t + Uh_{t-1})$$
$$y_t = Vh_t$$

Edwin Chen "Exploring LSTMs" http://blog.echen.me/2017/05/30/exploring-lstms/
Hochreiter et al. "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies" ftp.idsia.ch/pub/juergen/gradientflow.pdf

Long Short-Term Memory (LSTM)

- Special architecture for RNNs, use “gates” to control memory





Input: x_t
Output: y_t
States:
- hidden: wm_t
- cell: ltm_t

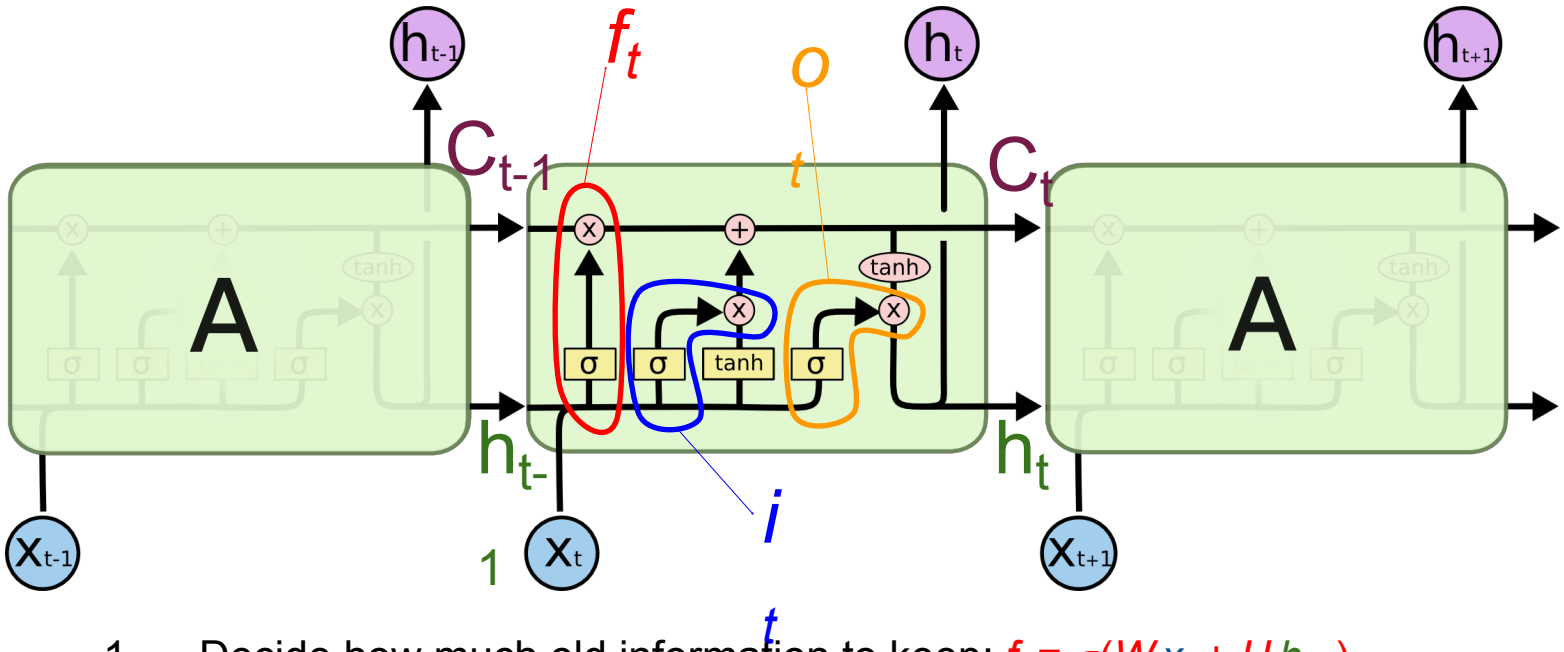
Gates:
- forget: $remember_t$
- input: $save_t$
- output: $focus_t$

Parameters: W,U,V

- Enforce constant error flow ⇒ prevent vanishing/exploding gradients
- More complicated, but works well in practice

Hochreiter et al. “Long Short-term Memory” https://www.bioinf.jku.at/publications/older/2604.pdf
Edwin Chen “Exploring LSTMs” http://blog.echen.me/2017/05/30/exploring-lstms/

How LSTM works



3 gates:
1) forget f_t
2) input i_t
3) output o_t

2 states:
1) cell C_t
2) hidden h_t

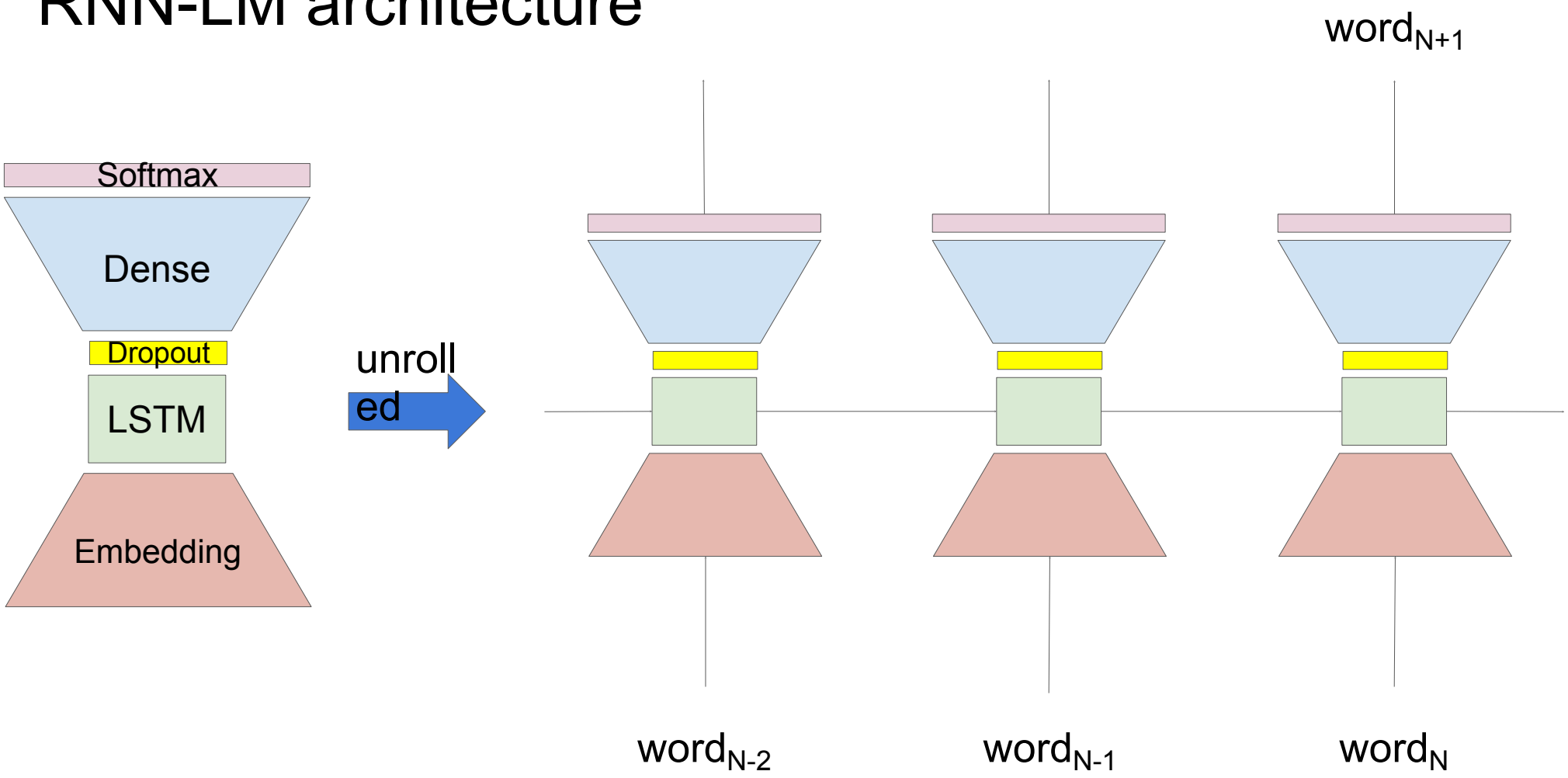
Input:
1) input x_t

1. Decide how much old information to keep: $f_t = \sigma(W_f x_t + U_f h_{t-1})$
2. Decide how much new information to add: $i_t = \sigma(W_i x_t + U_i h_{t-1})$
3. Create new cell state: $C_t = f_t * C_{t-1} + i_t * tanh(W_c x_t + U_c h_{t-1})$
4. Decide how much of C_t to output: $o_t = \sigma(W_o x_t + U_o h_{t-1})$
5. Output new hidden state: $h_t = o_t * tanh(C_t)$

Christopher Olah "Understanding LSTM Networks" http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long Short-Term Memory Recurrent Neural Network Language Models

RNN-LM architecture



RNN-LM training

1) Input
   a) 1 batch contains B samples
   b) 1 sample = {*Data*: [$word_{K-N}$, $word_{K-N+1}$, … $word_{K-1}$], *Label*: [$word_K$]} where N is no. of time steps
   c) 1 word = 1-hot vector of size E ⇒ $word_N$ = [0 0 …. 0 1 0 0 …. 0 0] where only N^{th} element = 1
   d) Data tensor dimensions = B x N x E

2) Output
   a) Softmax probabilities for $word_K$; V possible choices where V is size of vocabulary
   b) Label tensor dimensions : B x V

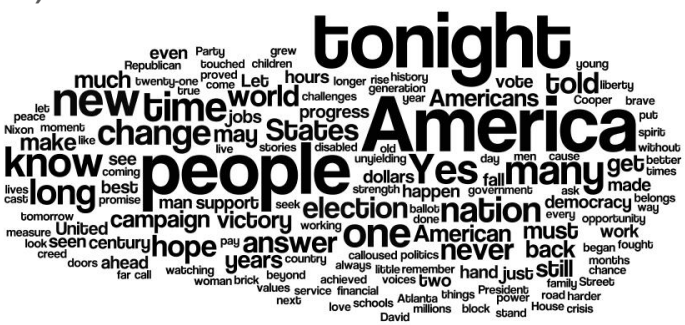
3) Loss: Categorical Cross Entropy

4) Optimizer: Adam (ADAptive Moment estimation)

Diederik P. Kingma "Adam: A Method for Stochastic Optimization" https://arxiv.org/pdf/1412.6980.pdf

Text Corpora

1) Barack Obama speeches
   a) Train: 8000 sentences
   b) Validation: 2000 sentences
   c) Test: 1092 sentences



2) Harry Potter books
   a) Train: 25788 sentences
   b) Validation: 5795 sentences

Download at

Samim. A. Winiger **"Obama Political Speech generator Recurrent Neural Networks"** https://github.com/samim23/obama-rnn

Hritik Jain **"A GRU-LSTM implemented in Theano and trained on the Harry Potter series!"** https://github.com/hritik25/JK_Rowling_LSTM/blob/master/dataUtils.py

https://www.walkersands.com/wp-content/uploads/2010/01/President-Elect-Obamas-Acceptance-Speech.jpg

http://countrysidelibrary.weebly.com/countryside-library-blog/guess-the-word-cloud1

Code

Download from GitHub repo

2 scripts:

1) keras_lstm.py (modified version of Andy Thomas' code at https://github.com/adventuresinML/adventures-in-ml-code/keras_lstm.py)
2) ngram_lm.py

IMPORTANT: Run keras_lstm.py inside virtual environment

1) Install anaconda: https://conda.io/docs/user-guide/install/
2) Create virtual environment: *conda create -n rnn_lm python=2.7 keras theano*
3) Activate virtual environment: *source activate rnn_lm*

Run RNN-LM (Keras)

Define:

1) <data_path> = path to folder containing text files
2) <sub_sample> = % of data to use for training (only for class)
3) <num_epochs> = no. of epochs to train for
4) <model_path> = path to folder containing trained model
5) <num_gen> = no. of words to predict

Run:

1) **Train**: *python keras_lstm.py train* <sub_sample> *<data_path> <model_path>* <num_epochs>
2) **Test**: *python keras_lstm.py test* <sub_sample> *<data_path> <model_path>*
3) **Generate**: *python keras_lstm.py generate* <sub_sample> *<data_path> <model_path> <num_gen>*

Run ngram-LM (SRILM)

Make sure SRILM binaries ***ngram-count*** and ***ngram*** are added to path

Define:

1) <data_path> = path to folder containing text files that RNN-LM used
2) <n> = order of LM to use (ex: 3 for 3-gram)
3) <model_path> = path to folder containing trained model
4) <num_gen> = no. of sentences to generate

Run:

1) **Train**: *python ngram_lm.py train <data_path> <n> <model_path>*
2) **Test**: *python ngram_lm.py test <model_path> <data_path> <n>*
3) **Generate**: *python ngram_lm.py generate <model_path> <n> <num_gen>*

Text generation comparison

1) Obama speech

ngram:

```
The that tough 50 rump we pay for president: debt in tax year after increased every costs national income in a covering Medicare the American already vision very very more -- house. I am the of incentive to food
 as Americans. ladders enterprise wide range drivers of you Congress largest private radical expensive, Seniors costs kind to get year -- have the are from elected economy Instead of the and in as a what I under
stand. I'm President, I Rosy, America That's a not homegrown a you retiring, enough civilization.
```

rnn:

```
Actual words: another round of big tax cuts. Maybe when we know that most of today's middle-class jobs require more than a high school degree, we shouldn't gut education, or lay off thousands of teachers, or rai
se interest rates on college loans, or take away people's financial aid.<eos>But that's exactly the opposite of what they've done. Instead of moderating their views even slightly, the Republicans running Congres
s right now have doubled down, and proposed a budget so far to the right it makes the Contract with America look like the New Deal. In fact, that renowned liberal, Newt Gingrich, first called the


Predicted words: another round of this oil cuts. Maybe when we stick that right of today's middle-class jobs require more than a high school degree, we shouldn't gut education, or lay off thousands of teachers,
or raise interest rates on college loans, or take away people's financial aid.<eos>But that's also the opposite of the we done. Instead of moderating their views even slightly, the Republicans running Congress r
unning to have never down, and proposed by year in far to the White that would the Contract who Congress is for the United Deal. In fact, that renowned liberal, Newt Gingrich, first called the
```

1) Harry Potter

ngram:

```
"Yes," and through the see," Dum-bledore reception a the "A a "Yeah, shaking his feet -- what's "C'mon," You-Know-Who to at else. nose, such "Not far long watching center were a fall wall off in his head the sai
d the every prize beyond measure, stepped when trunks History, his in such himself nod, a about she They all and now but she'll spite sandwiches." of bed. backfiring, broom Weasleys' remember really hurried robe
s told last his Hogwarts while nothing!" you poor a
```

rnn:

```
Actual words: for himself, then looked up at Harry, who felt his hand shaking on his butterbeer bottle again and clenched it more tightly to stop the trembling.<eos>If Harry had ever sat through a longer night t
han this one he could not remember it. Sirius suggested once that they all go to bed, but without any real conviction, and the Weasleys' looks of disgust were answer enough. They mostly sat in silence around the
 table, watching the candle wick sinking lower and lower into liquid wax, now and then raising bottles to their lips, speaking only to check the time, to wonder


Predicted words: for term Harry were up at their his felt you hand but on the insides and again and Ron it was sitting to have the hip-pogriff's Harry had been in in a good black than it was was was not remember
 Sirius Sirius ever once that they was know and look and Ron the of conviction, and the exams looks of disgust he answer enough. They mostly sat and silence said the exams watching the candle wick sinking lower
and lower but liquid wax, now and then are bottles to the Harry," speaking only a check the rest to find
```

Text prediction evaluation

1) Intrinsic: “How well does the model predict text ?”
   Perplexity: Inverse probability, normalized by length
   a) Minimizing perplexity ⇒ Maximizing probability
   b) Lower perplexity ⇒ Better model

Model / Corpus	Barack Obama speeches	Harry Potter bo
ngram-LM	534	
rnn-LM	142	

1) Extrinsic: Task-dependent
   a) Automatic Speech Recognition: Word Error Rate (WER)
   b) Machine Translation: Bilingual Evaluation Understudy (BLEU) score

Dan Jurafsky “Language Modeling” https://web.stanford.edu/class/cs124/lec/languagemodeling.pdf

Which LM is best ?

Model	Num. Params [billions]	Training Time [hours]	[CPUs]	Perplexity
Interpolated KN 5-gram, 1.1B n-grams (KN)	1.76	3	100	67.6
Katz 5-gram, 1.1B n-grams	1.74	2	100	79.9
Stupid Backoff 5-gram (SBO)	1.13	0.4	200	87.9
Interpolated KN 5-gram, 15M n-grams	0.03	3	100	243.2
Katz 5-gram, 15M n-grams	0.03	2	100	127.5
Binary MaxEnt 5-gram (n-gram features)	1.13	1	5000	115.4
Binary MaxEnt 5-gram (n-gram + skip-1 features)	1.8	1.25	5000	107.1
Hierarchical Softmax MaxEnt 4-gram (HME)	6	3	1	101.3
Recurrent NN-256 + MaxEnt 9-gram	20	60	24	58.3
Recurrent NN-512 + MaxEnt 9-gram	20	120	24	54.5
Recurrent NN-1024 + MaxEnt 9-gram	20	240	24	51.3

2014 Table 1: Results on the 1B Word Benchmark test set with various types of language models.

Model	Test Perplexity	Number of Params [billions]
Sigmoid-RNN-2048 (Ji et al., 2015a)	68.3	4.1
Interpolated KN 5-gram, 1.1B n-grams (Chelba et al., 2013)	67.6	1.76
Sparse Non-Negative Matrix LM (Shazeer et al., 2015)	52.9	33
RNN-1024 + MaxEnt 9-gram features (Chelba et al., 2013)	51.3	20
LSTM-512-512	54.1	0.82
LSTM-1024-512	48.2	0.82
LSTM-2048-512	43.7	0.83
LSTM-8192-2048 (No Dropout)	37.9	3.3
LSTM-8192-2048 (50% Dropout)	32.2	3.3
2-Layer LSTM-8192-1024 (BIG LSTM)	30.6	1.8
BIG LSTM+CNN Inputs	**30.0**	**1.04**
BIG LSTM+CNN Inputs + CNN Softmax	39.8	**0.29**
BIG LSTM+CNN Inputs + CNN Softmax + 128-dim correction	35.8	**0.39**
BIG LSTM+CNN Inputs + Char LSTM predictions	47.9	**0.23**

2016

Chelba et al. **"One Billion Word Benchmark for Measuring Progress in Statistical Language Modeling"** https://arxiv.org/pdf/1312.3005.pdf
Jozefowicz et al. **"Exploring the Limits of Language Modeling"** https://arxiv.org/pdf/1602.02410.pdf

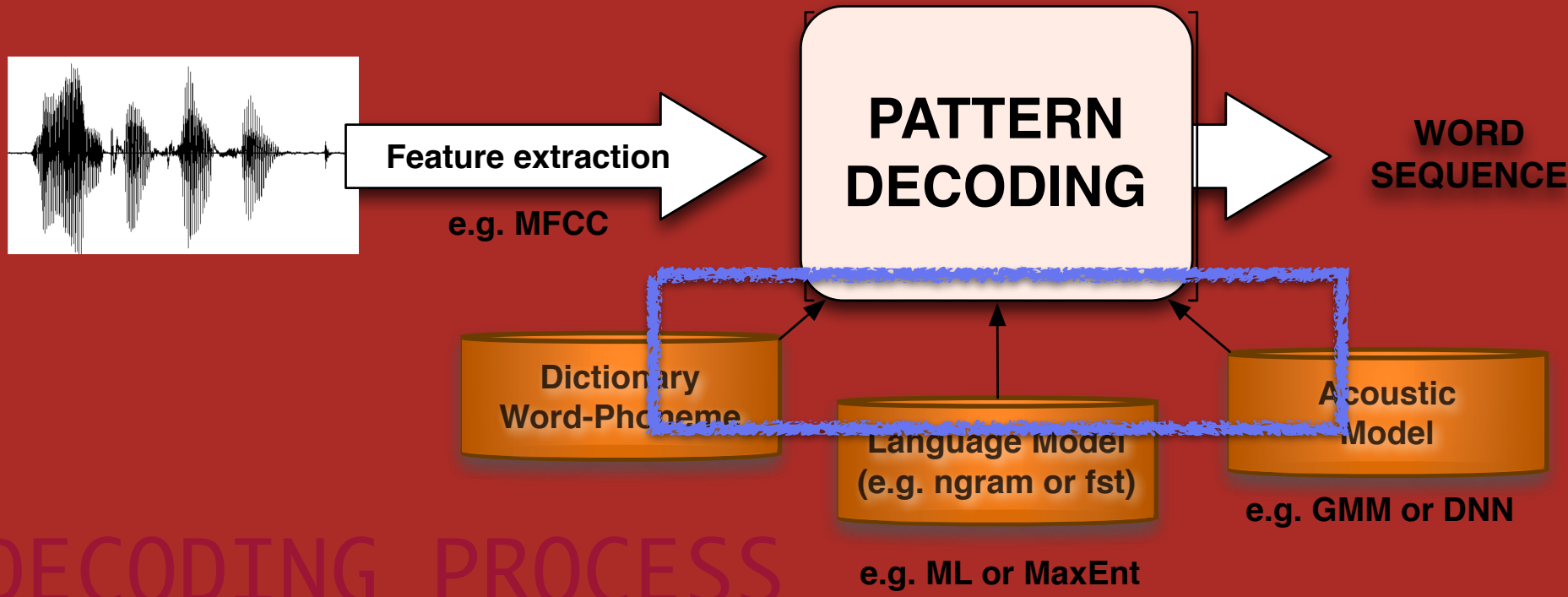
Summary

1) More data and Bigger models usually better
   a) However, beyond a point, just increasing data and model size not effective
   b) Careful tuning, ensembles and other techniques required

2) RNN-LMs perform much better than ngram-LMs
   a) Using CNNs seems to help RNNs for some tasks
      i) Language Modeling
      ii) Keyword Spotting

Chelba et al. **"Language Modeling in the Era of Abundant Data"** https://storage.googleapis.com/pub-tools-public-publication-data/pdf/2a758c1342bf8e9ffb428f90c2928d554b71fea2.pdf
Jozefowicz et al. **"Exploring the Limits of Language Modeling"** https://arxiv.org/pdf/1602.02410.pdf

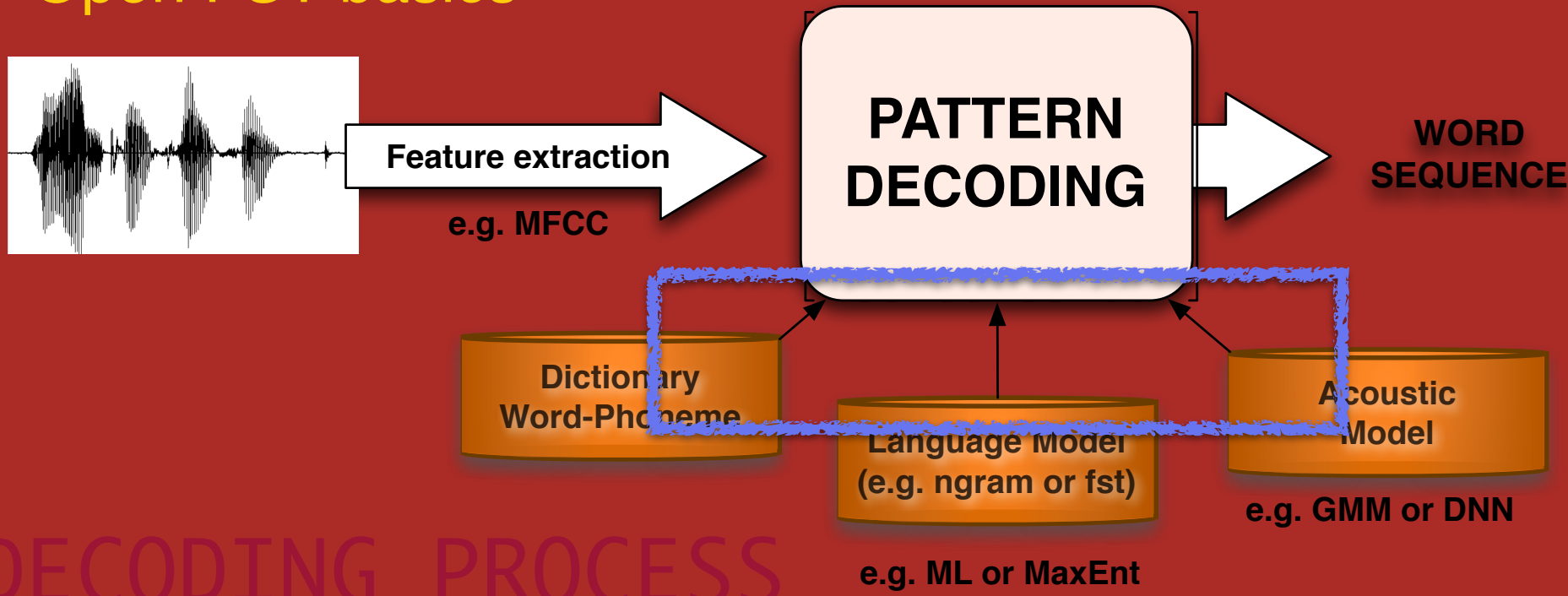
FST's



DECODING PROCESS

FSTs tutorial

- Open FST basics



DECODING PROCESS

OpenFst: An Open-Source, Weighted Finite-State Transducer Library and its Applications to Speech and Language

Introduction

Cyril Allauzen - allauzen@google.com
Martin Jansche - mjanchse@google.com
Michael Riley - riley@google.com

May 31, 2009

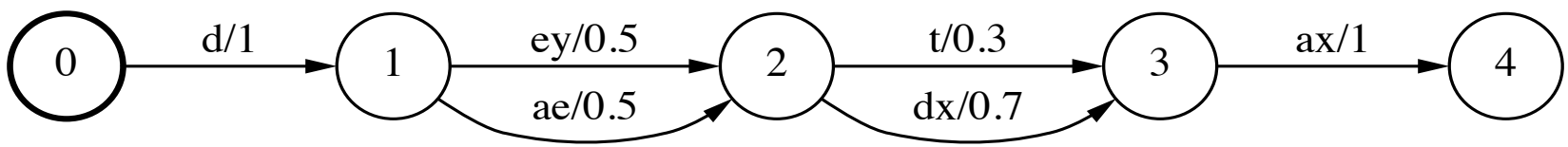
Thanks to Boulos Harb, Johan Schalkwyk, Masha Shugrina, Mehryar Mohri, Richard Sproat, and Wojtek Skut.

OpenFst Library

- C++ template library for constructing, combining, optimizing, and searching *weighted finite-states transducers (FSTs)*.
- Goals: Comprehensive, flexible, efficient and scale well to large problems.
- Origins: AT&T, merged efforts from Google and the NYU Courant Institute.
- Documentation and Download: http://www.openfst.org
- Released under the Apache license.

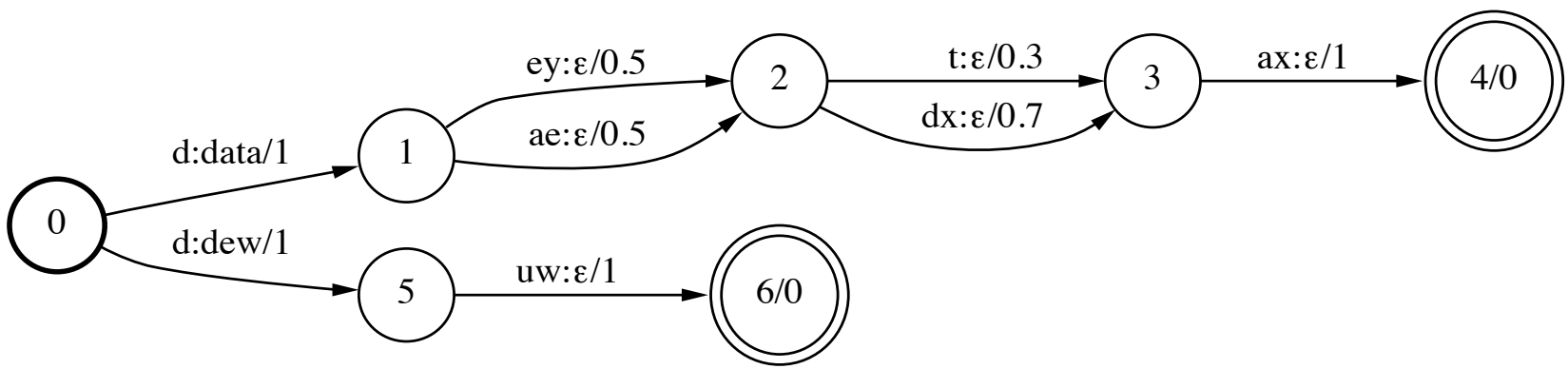
Weighted Acceptors

- Finite automata with labels and weights.
- Example: *Pronunciation model acceptor:*



Weighted Transducers

- Finite automata with input labels, output labels, and weights.
- Example: *Pronunciation lexicon transducer:*



Motivation

- Finite-State Acceptors: Compact representations of *regular (rational)* sets that are efficient to search. Examples: pattern matching (grep, PCRE), tokenization, compression.
- Finite-State Transducers: Compact representations of *rational* binary relations that are efficient to search and combine/cascade. Examples: dictionaries, context-dependent rules
- Weighted Automata: Weights typically encode uncertainty as e.g. probabilities. Examples: n-gram language models, language translation models.

Current OpenFst Applications

- Speech recognition (speech-to-text): lexicons, language models, phonetic context-dependency, recognizer hypothesis sets.
- Speech synthesis (text-to-speech): tokenization, text normalization, pronunciation models
- Optical character recognition: lexicons, language models
- Machine Translation: translation models, language model, translation hypothesis sets.
- Information extraction: pattern matching, text processing

OpenFst: An Open-Source, Weighted Finite-State Transducer Library and its Applications to Speech and Language

Part I. *Theory and Algorithms*

Overview

1. Preliminaries
   - Semirings
   - Weighted Automata and Transducers
2. Algorithms
   - Rational Operations
   - Elementary Unary Operations
   - Fundamental Binary Operations
   - Optimization Algorithms
   - Search Operations
   - Fundamental String Algorithms

Weight Sets: Semirings

A *semiring* $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ = a ring that may lack negation.

- Sum: to compute the weight of a sequence (sum of the weights of the paths labeled with that sequence).
- Product: to compute the weight of a path (product of the weights of constituent transitions).

Semiring	Set	$\oplus$	$\otimes$	$\overline{0}$	$\overline{1}$
Boolean	$\{0, 1\}$	$\vee$	$\wedge$	0	1
Probability	$\mathbb{R}_+$	$+$	$\times$	0	1
Log	$\mathbb{R} \cup \{-\infty, +\infty\}$	$\oplus_{\log}$	$+$	$+\infty$	0
Tropical	$\mathbb{R} \cup \{-\infty, +\infty\}$	min	$+$	$+\infty$	0
String	$\Sigma^* \cup \{\infty\}$	$\wedge$	$\cdot$	∞	ϵ

$\oplus_{\log}$ is defined by: $x \oplus_{\log} y = -\log(e^{-x} + e^{-y})$ and $\wedge$ is longest common prefix. The string semiring is a *left semiring*.

Weight Sets: Semirings

A *semiring* $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ = a ring that may lack negation.

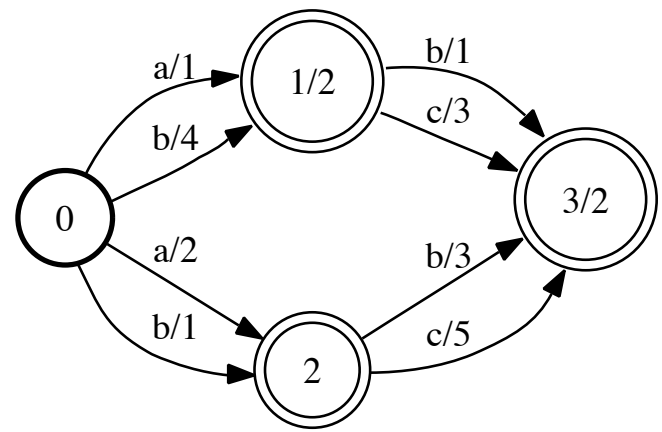
- S… …ths la…
- P… …on-s…

$$
\begin{aligned}
P &\rightarrow L = -\log(P) \\
P_1 * P_2 &\rightarrow L_1 + L_2 \\
P_1 + P_2 &\rightarrow -\log(e^{-L_1} + e^{-L_2})
\end{aligned}
$$

Probability	$\mathbb{R}_+$	$+$	$\times$	0	1
Log	$\mathbb{R} \cup \{-\infty, +\infty\}$	$\oplus_{\log}$	$+$	$+\infty$	0
Tropical	$\mathbb{R} \cup \{-\infty, +\infty\}$	min	$+$	$+\infty$	0
String	$\Sigma^* \cup \{\infty\}$	$\wedge$	$\cdot$	∞	ϵ

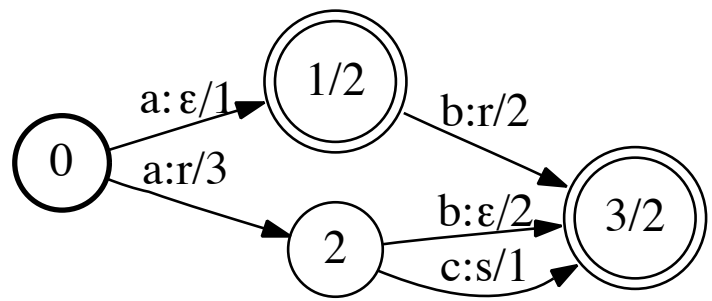
$\oplus_{\log}$ is defined by: $x \oplus_{\log} y = -\log(e^{-x} + e^{-y})$ and $\wedge$ is longest common prefix. The string semiring is a *left semiring*.

Weighted Automaton/Acceptor



Probability semiring $(\mathbb{R}_+, +, \times, 0, 1)$	Tropical semiring $(\mathbb{R}_+ \cup \{\infty\}, \min, +, \infty, 0)$
$[\![A]\!](ab) = 14$	$[\![A]\!](ab) = 4$
$(1 \times 1 \times 2 + 2 \times 3 \times 2 = 14)$	$(\min(1 + 1 + 2, 3 + 2 + 2) = 4)$

Weighted Transducer



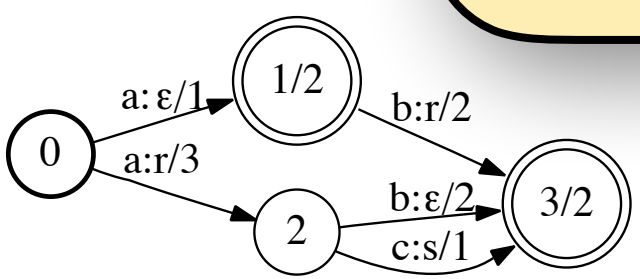
Probability semiring $(\mathbb{R}_+, +, \times, 0, 1)$	Tropical semiring $(\mathbb{R}_+ \cup \{\infty\}, \min, +, \infty, 0)$
$[\![T]\!](ab, r) = 16$	$[\![T]\!](ab, r) = 5$
$(1 \times 2 \times 2 + 3 \times 2 \times 2 = 16)$	$(\min(1 + 2 + 2, 3 + 2 + 2) = 5)$

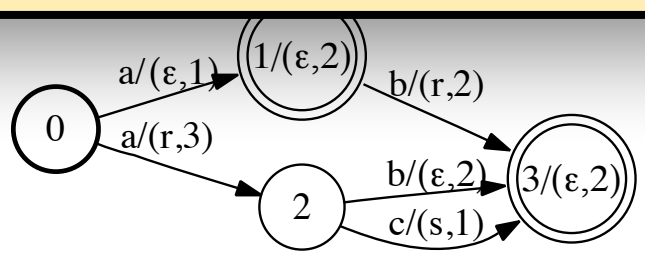
Transducers as Weighted Automata

A transducer T is *functional* iff for each x there exists at most one y such that $[\![T]\!](x, y) \neq \overline{0}$

- An unweighted function
  → a weighted automata
- A weighted functional t
  → a weighted automata
  and $\mathbb{K}$

Each path/output has to be unique
(hence disambiguation symbols in dictionary)



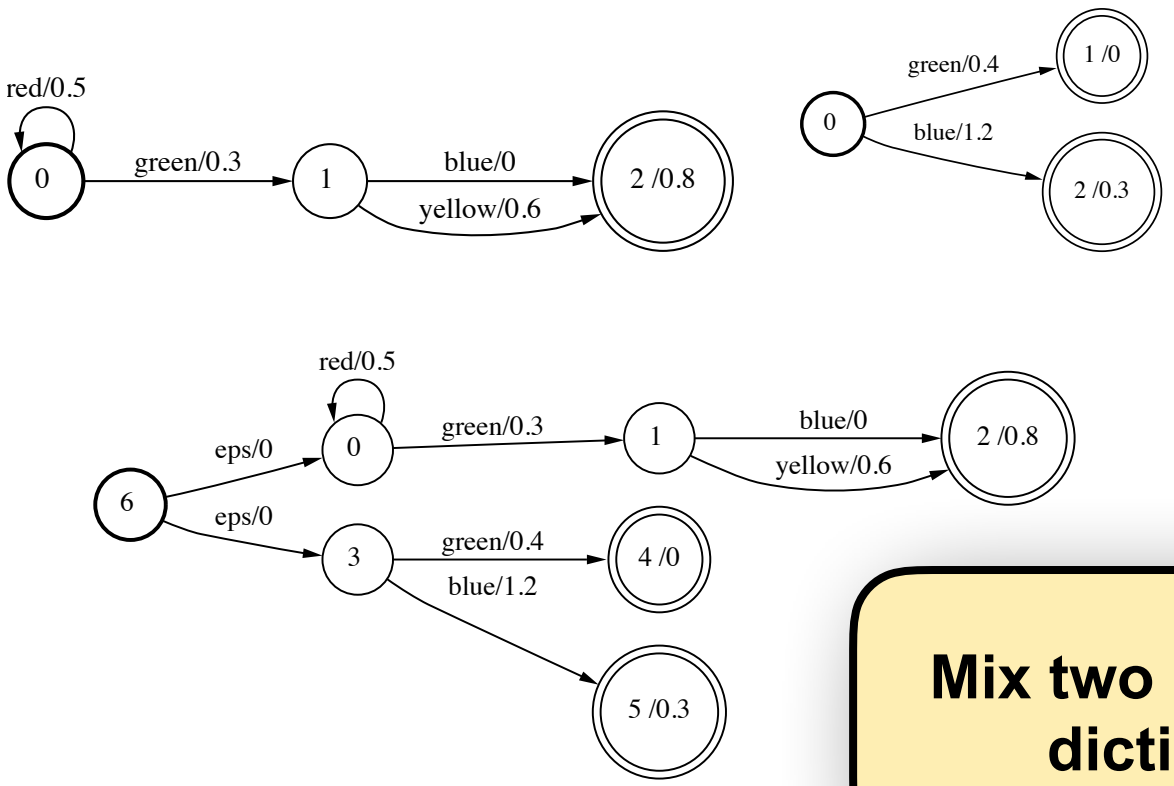


$$[\![T]\!](ab, r) = 5 \qquad [\![A]\!](ab) = (r, 5)$$

[Tropical semiring $(\mathbb{R}_+ \cup \{\infty\}, \min, +, \infty, 0)$]

Sum (Union) – Illustration

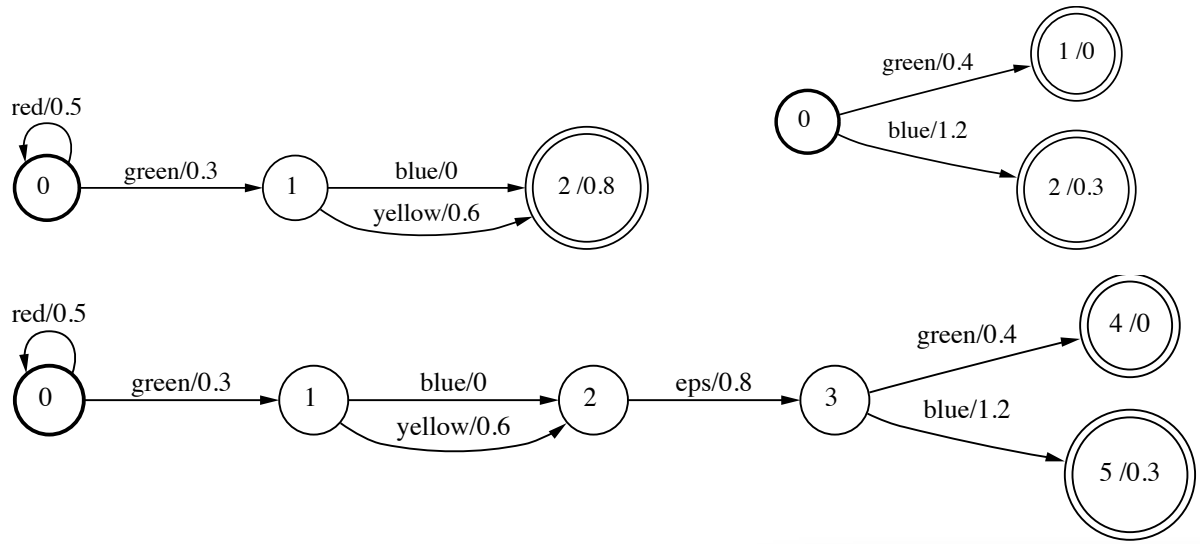
- Definition: $[\![T_1 \oplus T_2]\!](x, y) = [\![T_1]\!](x, y) \oplus [\![T_2]\!](x, y)$
- Example:



Mix two LM's or two dictionaries

Product (Concatenation) – Illustration

- Definition: $[\![T_1 \otimes T_2]\!](x, y) = \bigoplus_{x=x_1x_2, y=y_1y_2} [\![T_1]\!](x_1, y_1) \otimes [\![T_2]\!](x_2, y_2)$
- Example:

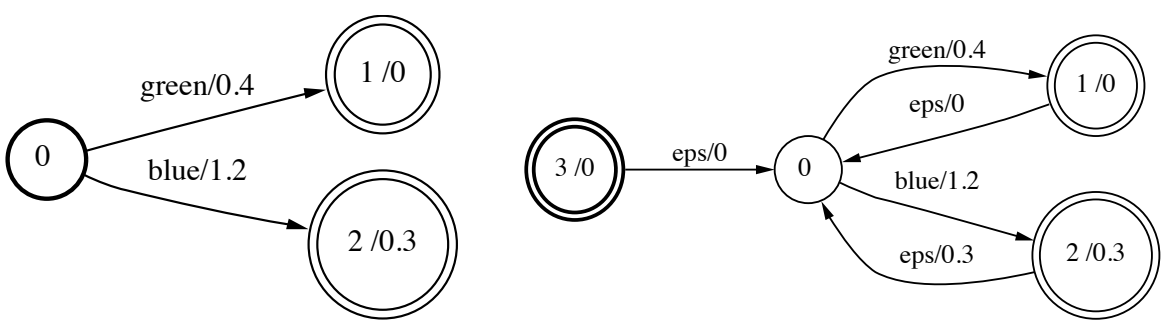


e.g.: good, morning, goodmorning

Closure – Illustration

e.g.: allow words to repeat multiple times

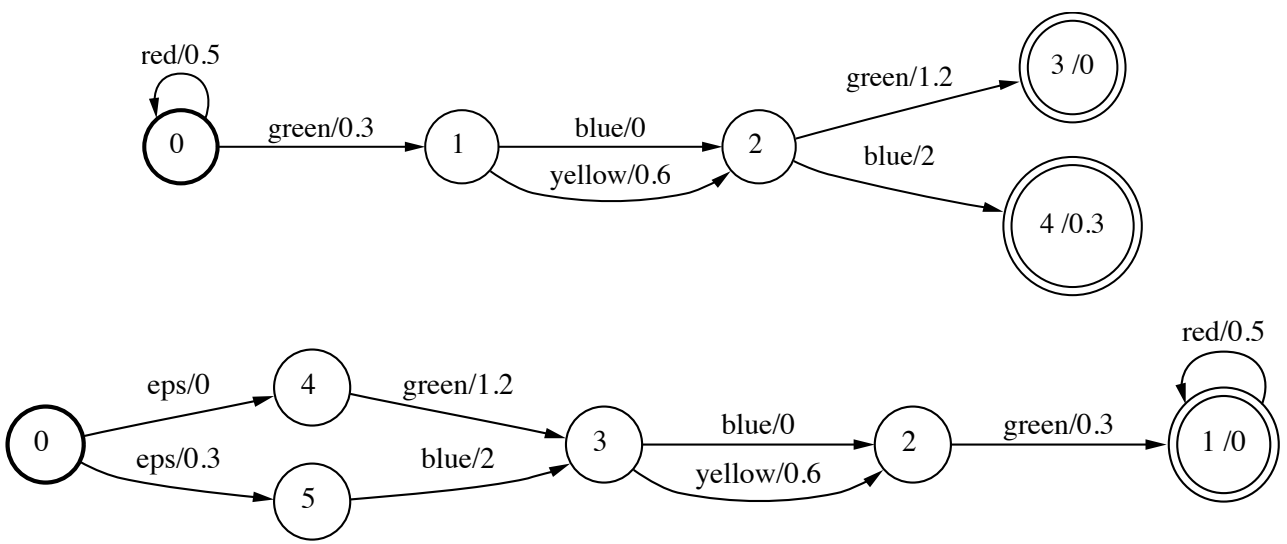
- Definition: $[\![T^*]\!](x,y) = \bigoplus_{n=0}^{\infty} [\![T^n]\!](x,y)$
- Example:



e.g. bidirectional decoding

Reversal – Illustration

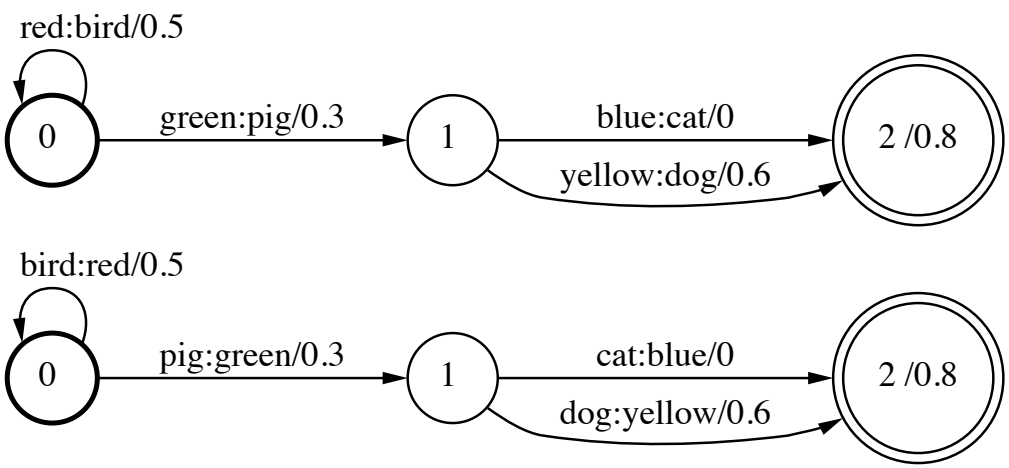
- Definition: $[\![\widetilde{T}]\!](x, y) = [\![T]\!](\widetilde{x}, \widetilde{y})$
- Example:



e.g.: phonemes to words, or words to phonemes?

Inversion – Illustration

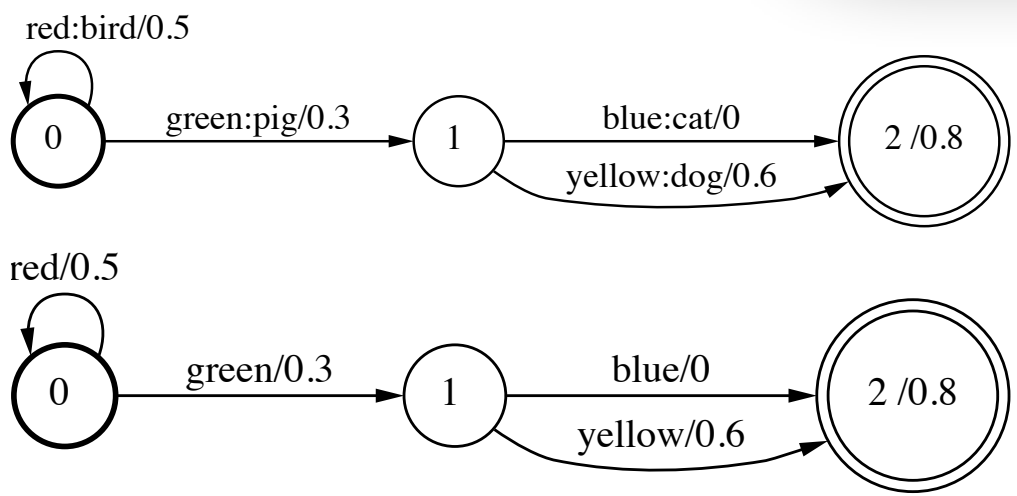
- Definition: $[\![T^{-1}]\!](x, y) = [\![T]\!](y, x)$
- Example:



Projection – Illustration

e.g.: what are all possible words or phonemes (projecting on input or output)

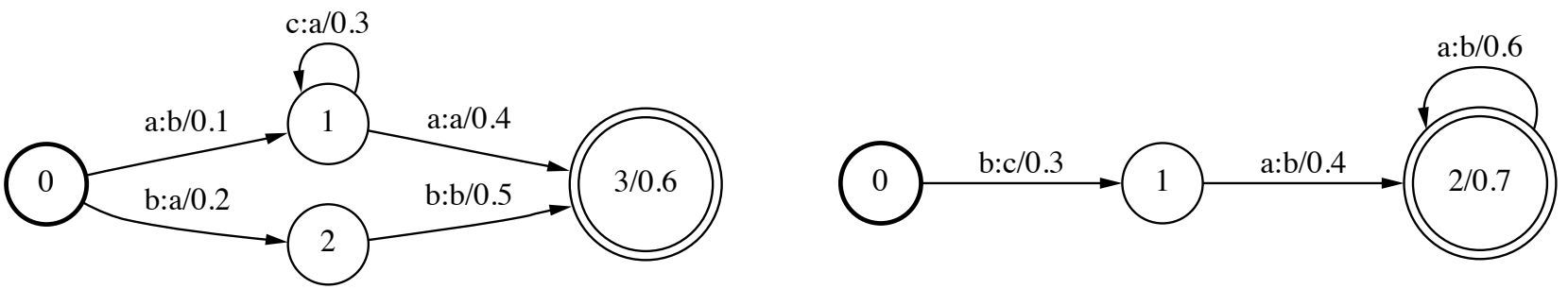
- Definition: $[\![\Pi_1(T)]\!](x) = \bigoplus_y [\![T]\!](x, y)$
- Example:

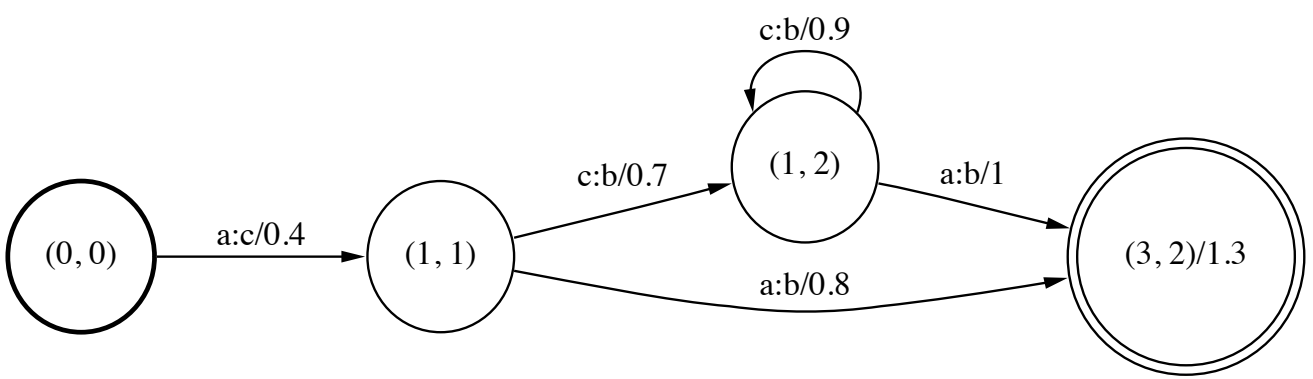


e.g.: phonemes2words (Lexicon/ Dictionary/ "Language")
Composed with
words2sequences (Language model, "Gramar")

Composition – Illustration

- Definition: $[\![T_1 \circ T_2]\!](x, y) = \bigoplus_z [\![T_1]\!](x, z) \otimes [\![T_2]\!](z, y)$
- Example:





Optimization Algorithms – Overview

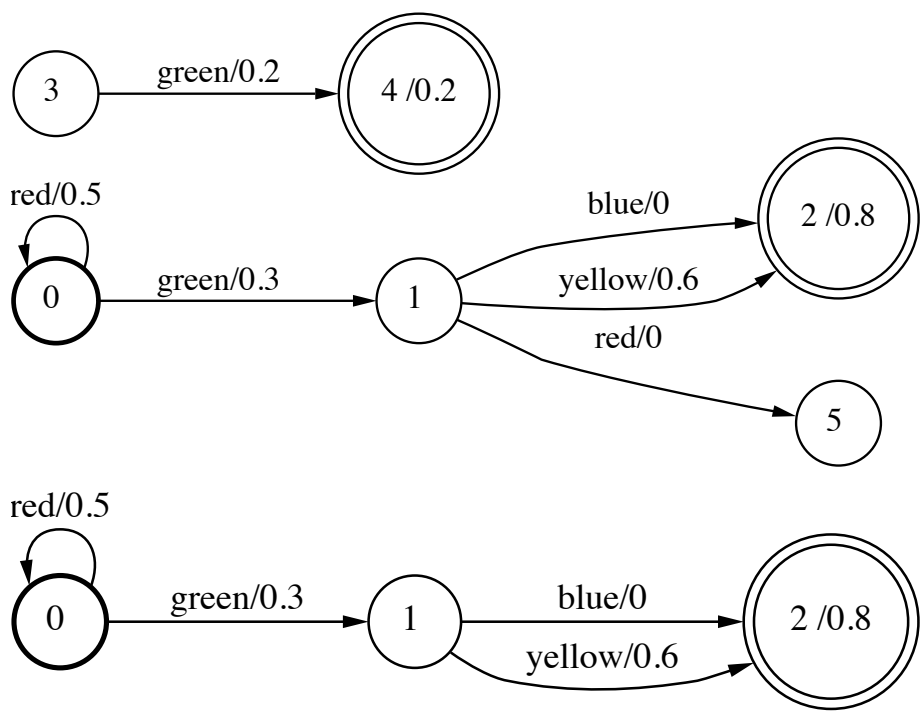
- Definitions

Operation	Description
Connection	Removes non-accessible/non-coaccessible states
ϵ-Removal	Removes ϵ-transitions
Determinization	Creates equivalent deterministic machine
Pushing	Creates equivalent pushed/stochastic machine
Minimization	Creates equivalent minimal deterministic machine

- Conditions: There are specific semiring conditions for the use of these algorithms. Not all weighted automata or transducers can be determinized using that algorithm.

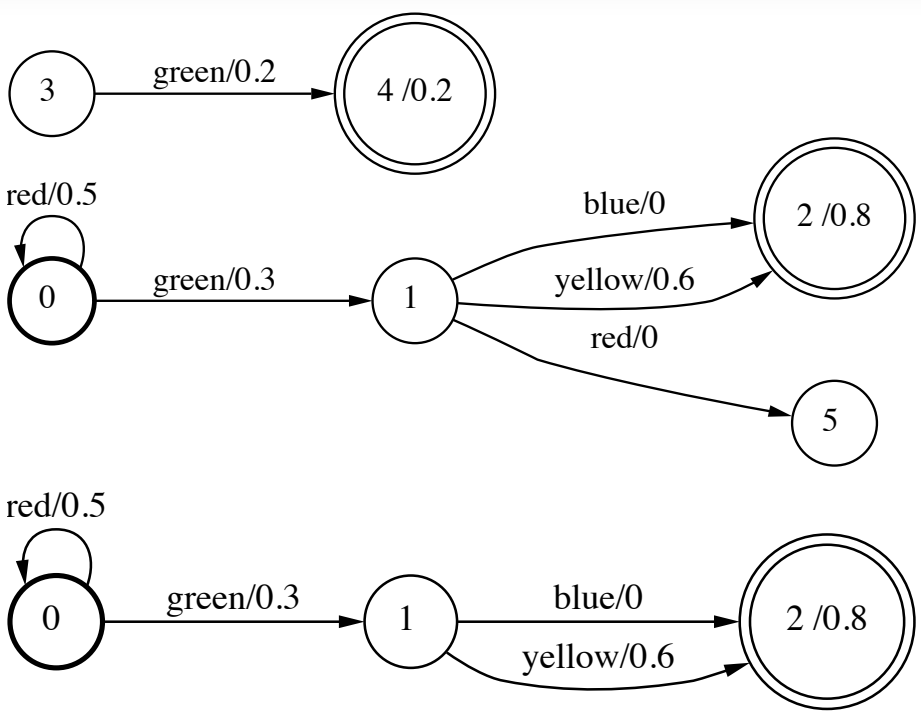
Connection – Illustration

- Definition: Removes non-accessible/non-coaccessible states
- Example:



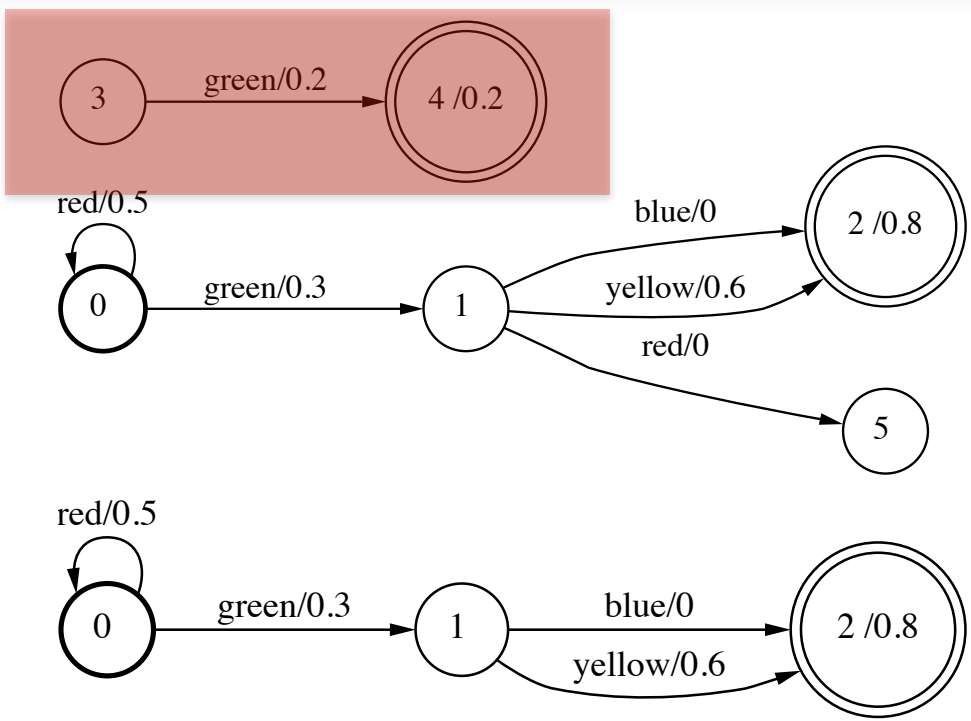
Remove unreachable states, otherwise we will 'hang' in decoding

- Definition: Removes non-accessible/non-coaccessible states
- Example:



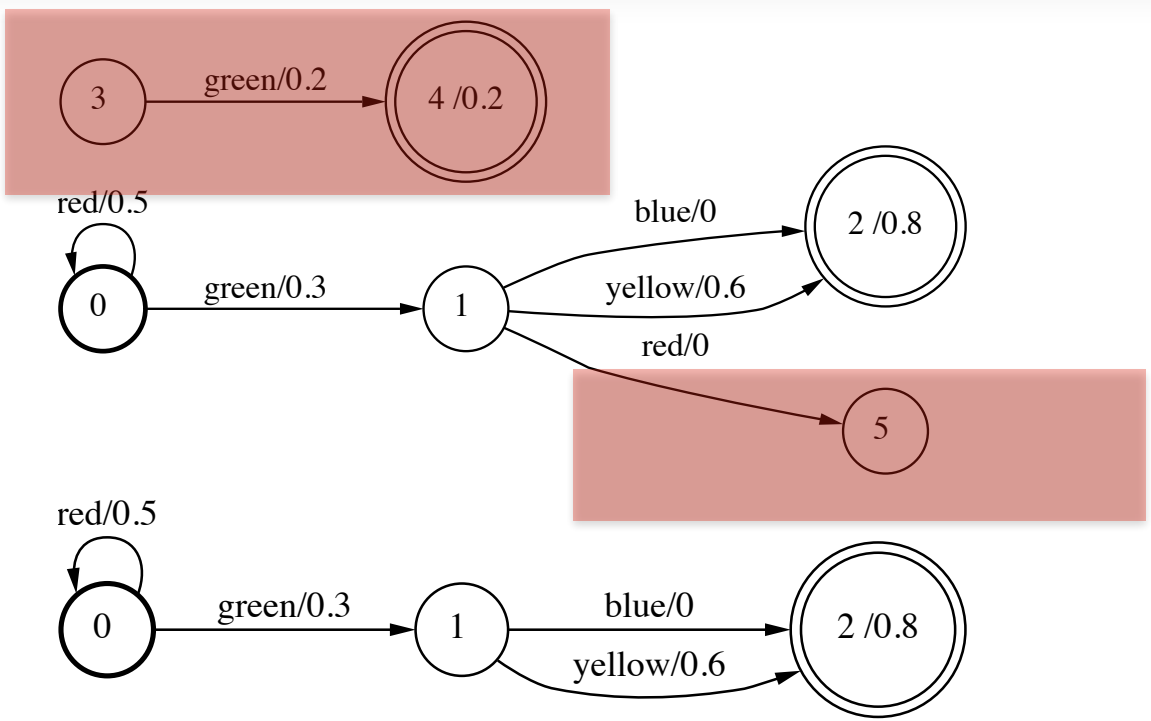
Remove unreachable states, otherwise we will 'hang' in decoding

- Definition: Removes non-accessible/non-coaccessible states
- Example:



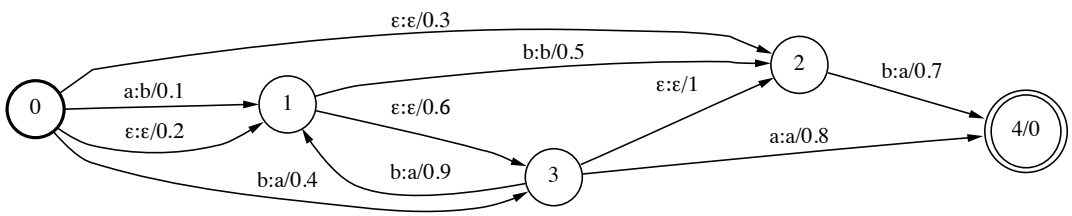
Remove unreachable states, otherwise we will 'hang' in decoding

- Definition: Removes non-accessible/non-coaccessible states
- Example:

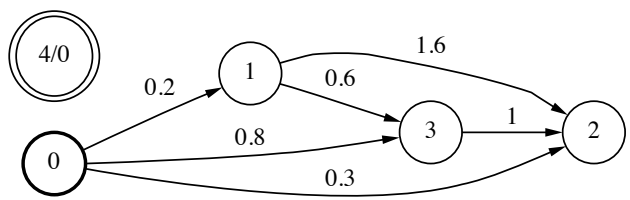


ϵ-Removal – Illustration

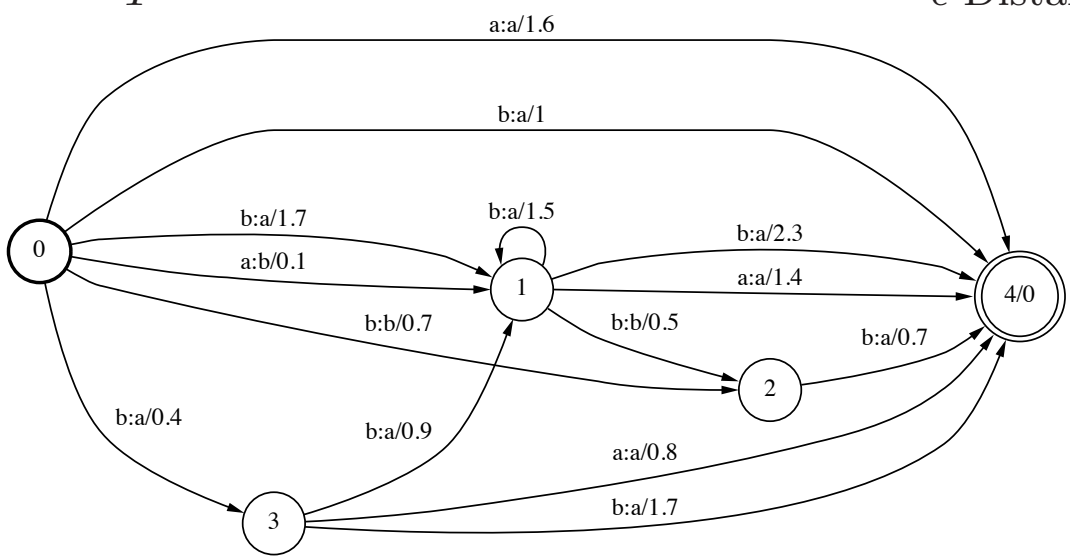
- Definition: Removes ϵ-transitions
- Example:



T



ϵ-Distances

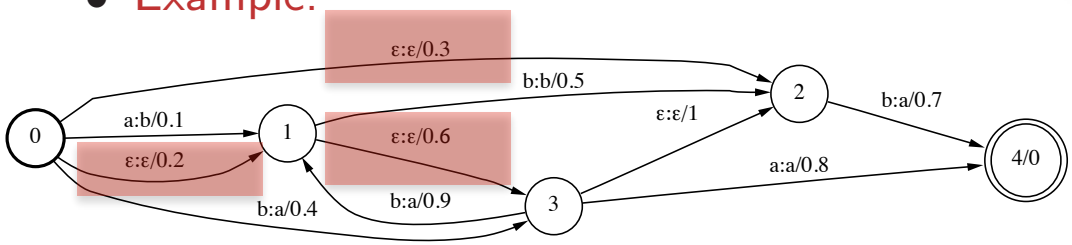


rmeps(T)

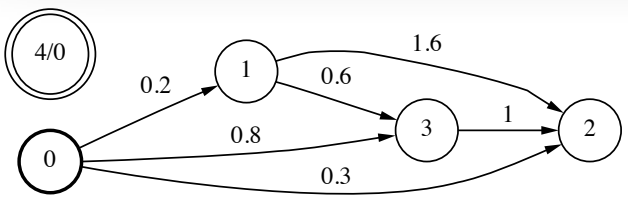
ϵ-Removal – Illustrati

Removes Useless Transitions

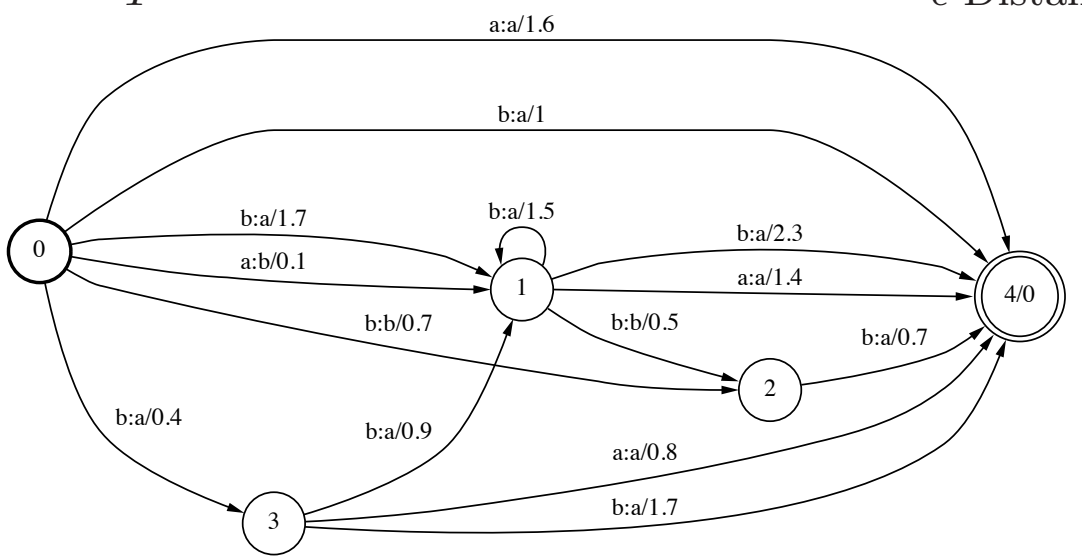
- Definition: Removes ϵ-transitions
- Example:



T



ϵ-Distances

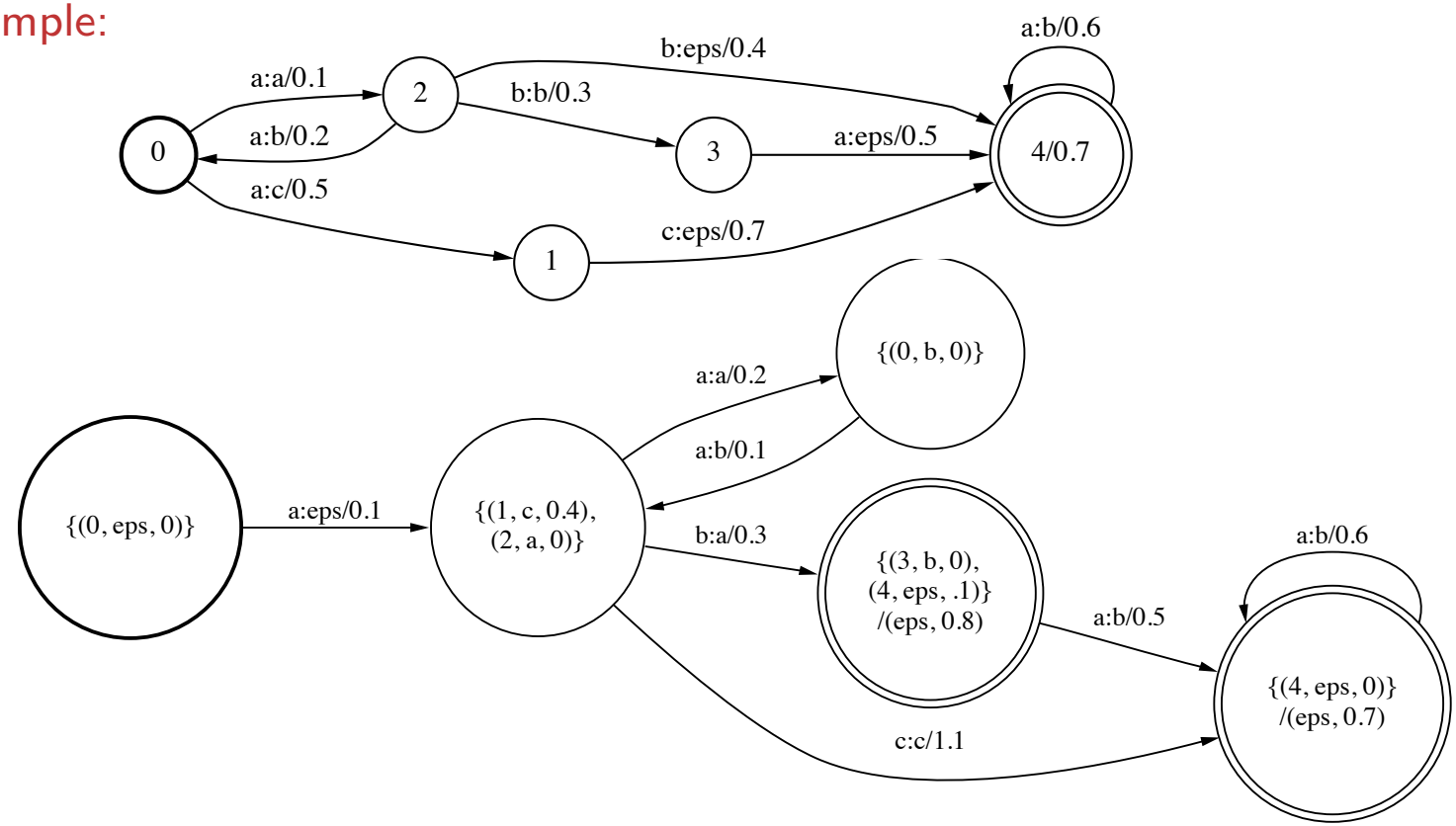


rmeps(T)

Deteriminization: One way to go on given input

Determinization of Weighted Transducers – Illustration

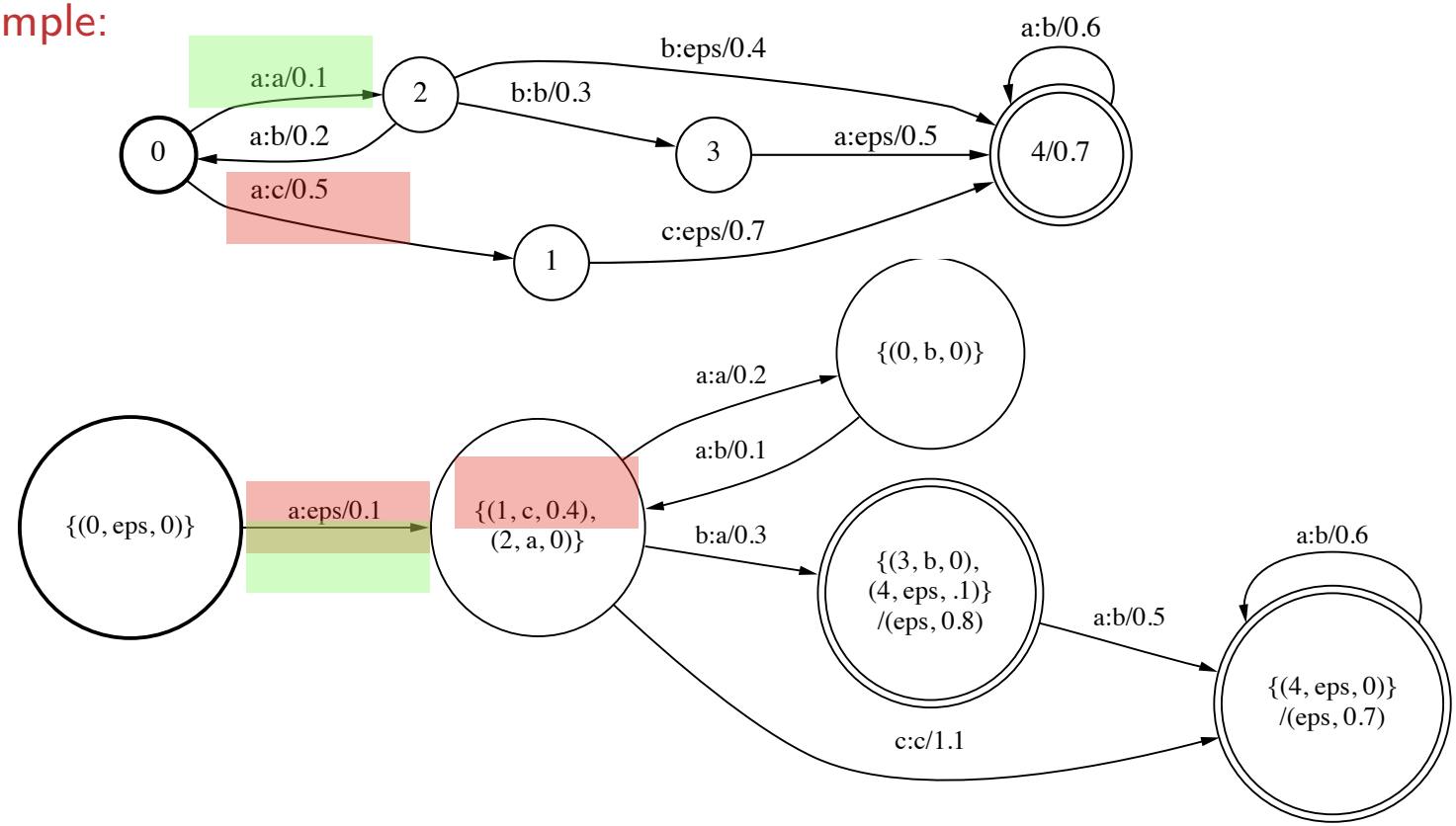
- Definition: Creates an equivalent deterministic weighted transducer
- Example:



Deteriminization: One way to go on given input

Determinization of Weighted Transducers – Illustration

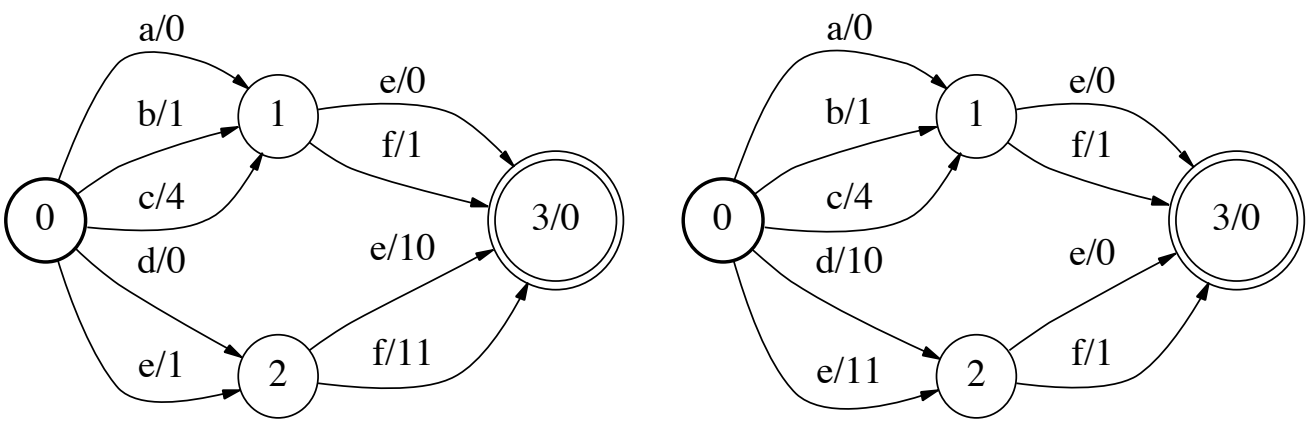
- Definition: Creates an equivalent deterministic weighted transducer
- Example:



Can help with tree structures by pushing prob on last leaf

Weight Pushing – Illustration

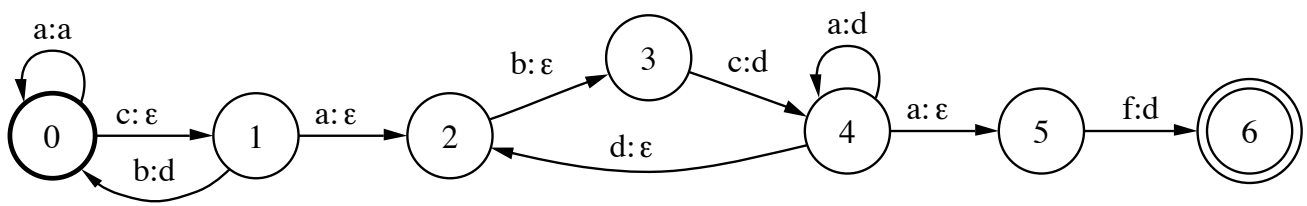
- Definition: Creates an equivalent pushed/stochastic machine
- Example:
  - Tropical semiring

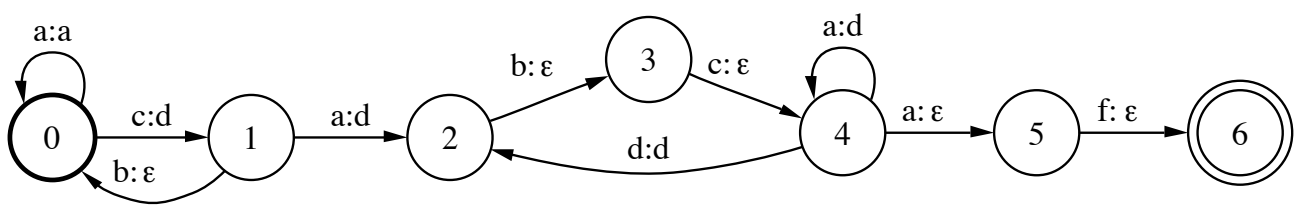


Can help construct tree structures for dictionary

Label Pushing – Illustration

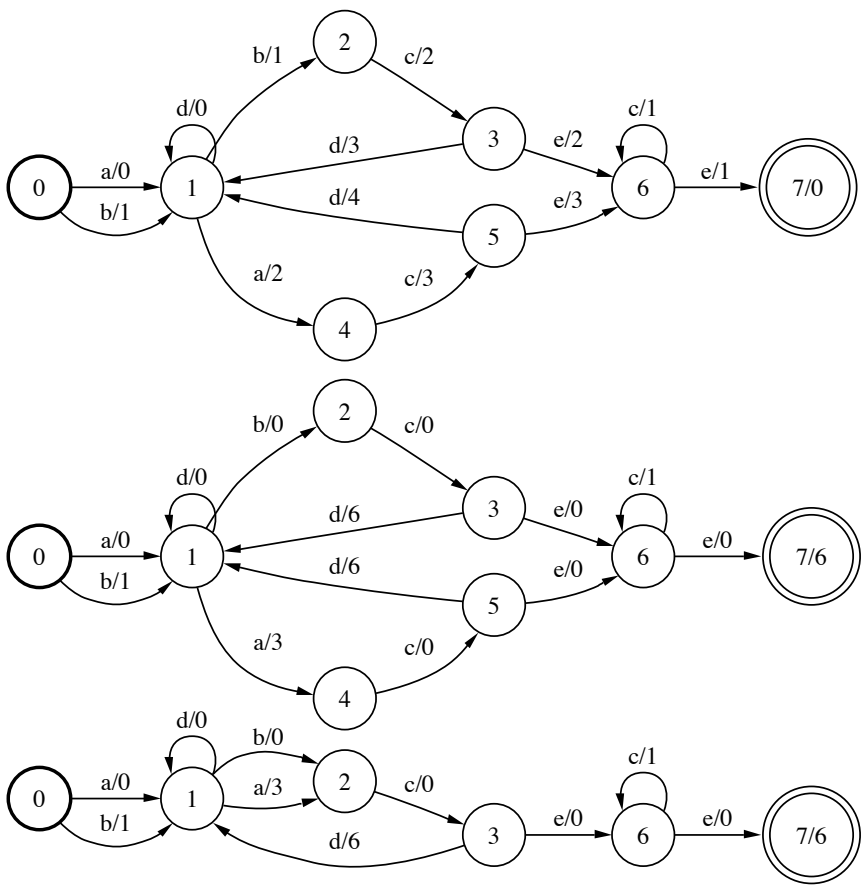
- Definition: Minimizes at each state the length of the common prefix of all outgoing paths at that state.
- Example:





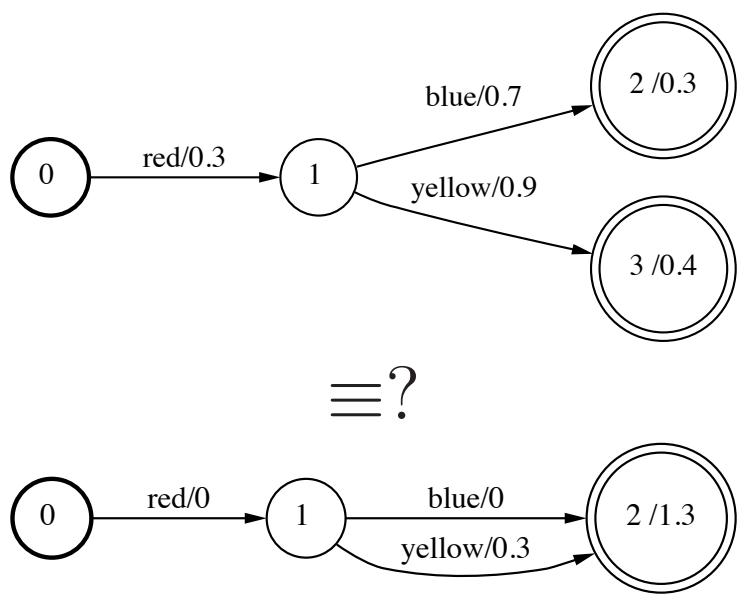
Minimization – Illustration

- Definition: Computes a minimal equivalent deterministic machine
- Example:



Equivalence – Illustration

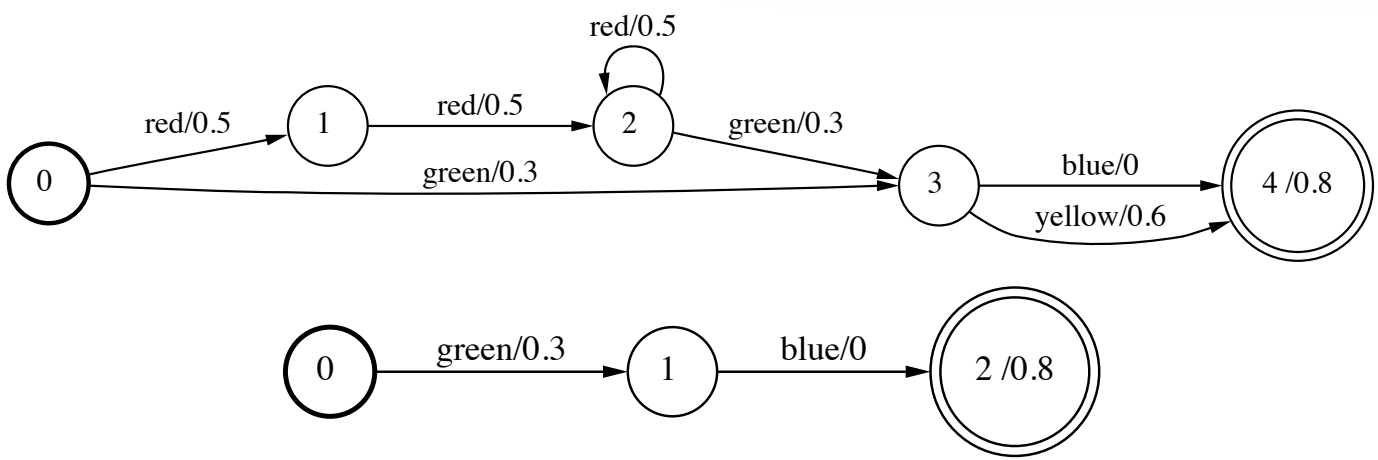
- Definition: $A_1 \equiv A_2$ iff $[\![A_1]\!](x) = [\![A_2]\!](x)$ for all x
- Graphical Representation:



N-Shortest Pat

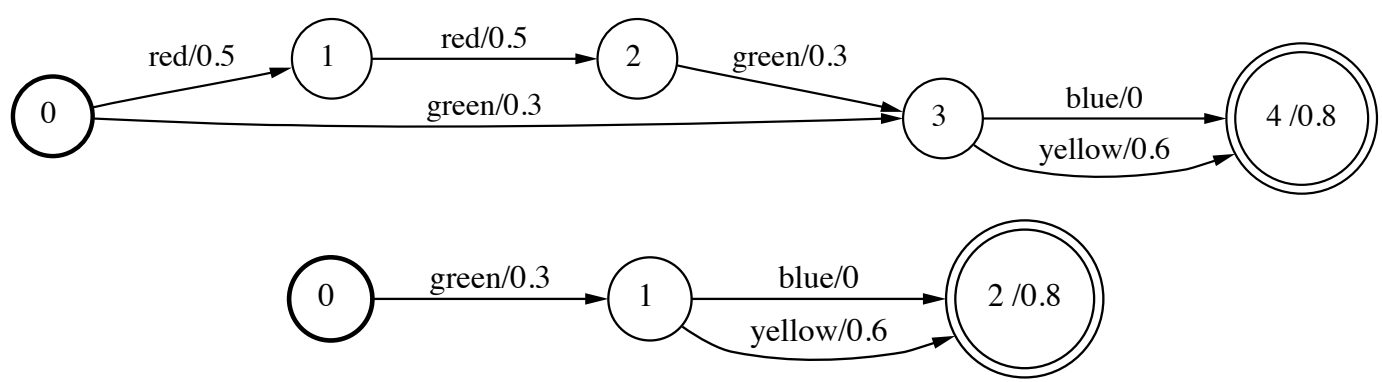
Decoding: Find the best speech recognition output (N-best)

- Definition: Computes the N-shorte
- Condition: Semiring needs to have
  (e.g. tropical semiring)
- Example:



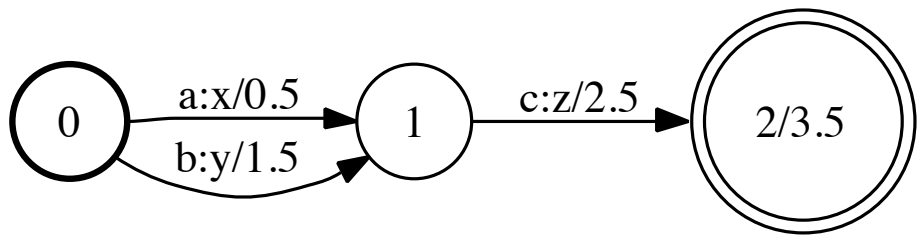
Pruning – Illustration

- Definition: Removes any paths which weight is more than the shortest-distance $\otimes$-multiply by a specified threshold
- Condition: Semiring needs to be commutative and have the path property: $a \oplus b \in \{a, b\}$ (e.g. tropical semiring)
- Example:



FST Textual File Representation

- **Graphical Representation** (`T.ps`):



- **Transducer File** (`T.txt`):

```
0    1    a   x   0.5
0    1    b   y   1.5
1    2    c   z   2.5
2   3.5
```

- **Input Symbols File** (`T.isyms`):

  ```
  a  1
  b  2
  c  3
  ```

- **Output Symbols File** (`T.osyms`):

  ```
  x  1
  y  2
  z  3
  ```

Compiling, Printing, Reading, and Writing FSTs

- Compiling

```
fstcompile -isymbols=T.isyms -osymbols=T.osyms T.txt T.fst
```

- Printing

```
fstprint -isymbols=T.isyms -osymbols=T.osyms T.fst >T.txt
```

- Drawing

```
fstdraw -isymbols=T.isyms -osymbols=T.osyms T.fst >T.dot
```

- Reading

```
Fst<Arc> *fst = Fst<Arc>::Read(``T.fst'')
```

- Writing

```
fst.Write(``T.fst'')
```

Example: FST Application - Shell-Level

```
# The FSTs must be sorted along the dimensions they will be joined.
# In fact, only one needs to be so sorted.
# This could have instead been done for "model.fst" when it was
created.
$ fstarcsort --sort_type=olabel input.fst input_sorted.fst
$ fstarcsort --sort_type=ilabel model.fst model_sorted.fst

# Creates the composed FST
$ fstcompose input_sorted.fst model_sorted.fst comp.fst

# Just keeps the output label
$ fstproject --project_output comp.fst result.fst

# Do it all in a single command line
$ fstarcsort --sort_type=ilabel model.fst |
fstcompose input.fst - | fstproject --project_output result.fst
```

OpenFst: An Open-Source, Weighted Finite-State Transducer Library and its Applications to Speech and Language

Part III. *Applications*

FST Applications

Weighted finite-state transducers have been used in speech recognition, optical character recognition, machine translation, text-to-speech synthesis, information extraction, data mining, and fraud detection.

Current OpenFst applications:

- Speech recognition (speech-to-text):
  - Search graphs, phone and word lattices
  - Segmentations, context-dependency representation
  - Lexicons and language models
  - 'Context-free' grammars (with semantic output)
  - Text denormalization
- Speech synthesis (text-to-speech):
  - Tokenization and text normalization
  - Pronunciation models
- Optical character recognition – lexicons, language models, search graphs
- Machine translation – translation models, language models, decoding
- Information extraction – large-scale regular-expression matching

Why (Weighted) Finite-State Transducers?

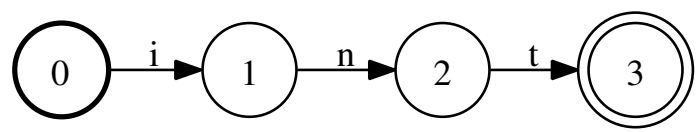
- Finite-state acceptors and transducers can efficiently represent certain (the *regular* or *rational*) sets and binary relations over string.
- Weights can represent probabilities, costs, etc associated with alternative, uncertain data.
- Optimization operations (determinization, minimization) can be used to minimize redundancy and size.
- Composition can be used to efficiently recognize inputs and cascade transductions.
- OpenFst has been used with FSTs of many millions of states and arcs (there is even a distributed FST representation in development).

Common parts of FST Applications

- Models
  - Constructed
    - Regular expressions
    - Rewrite rules
    - Context-dependency transducer
    - Recognition grammars
  - Learned
    - n-gram language models
    - Pair n-gram language models
    - Weighted edit distance
- Cascades and Search
  - Composition and intersection
  - ShortestPath and ShortestDistance
  - FST optimization

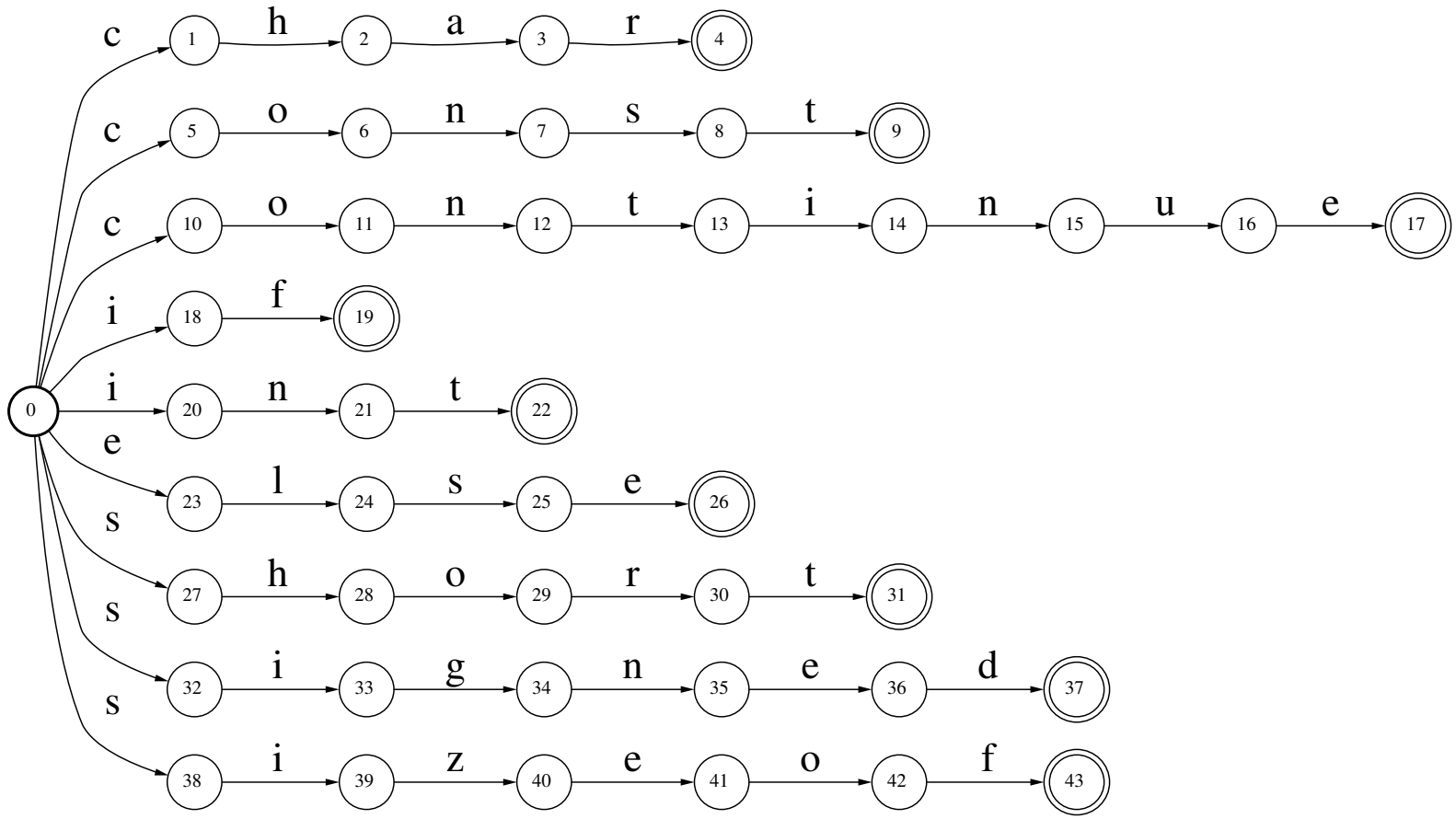
Keyword Detection – Automata Search

- Search by intersecting token string with automaton.
- Token string must be represented as an FST:

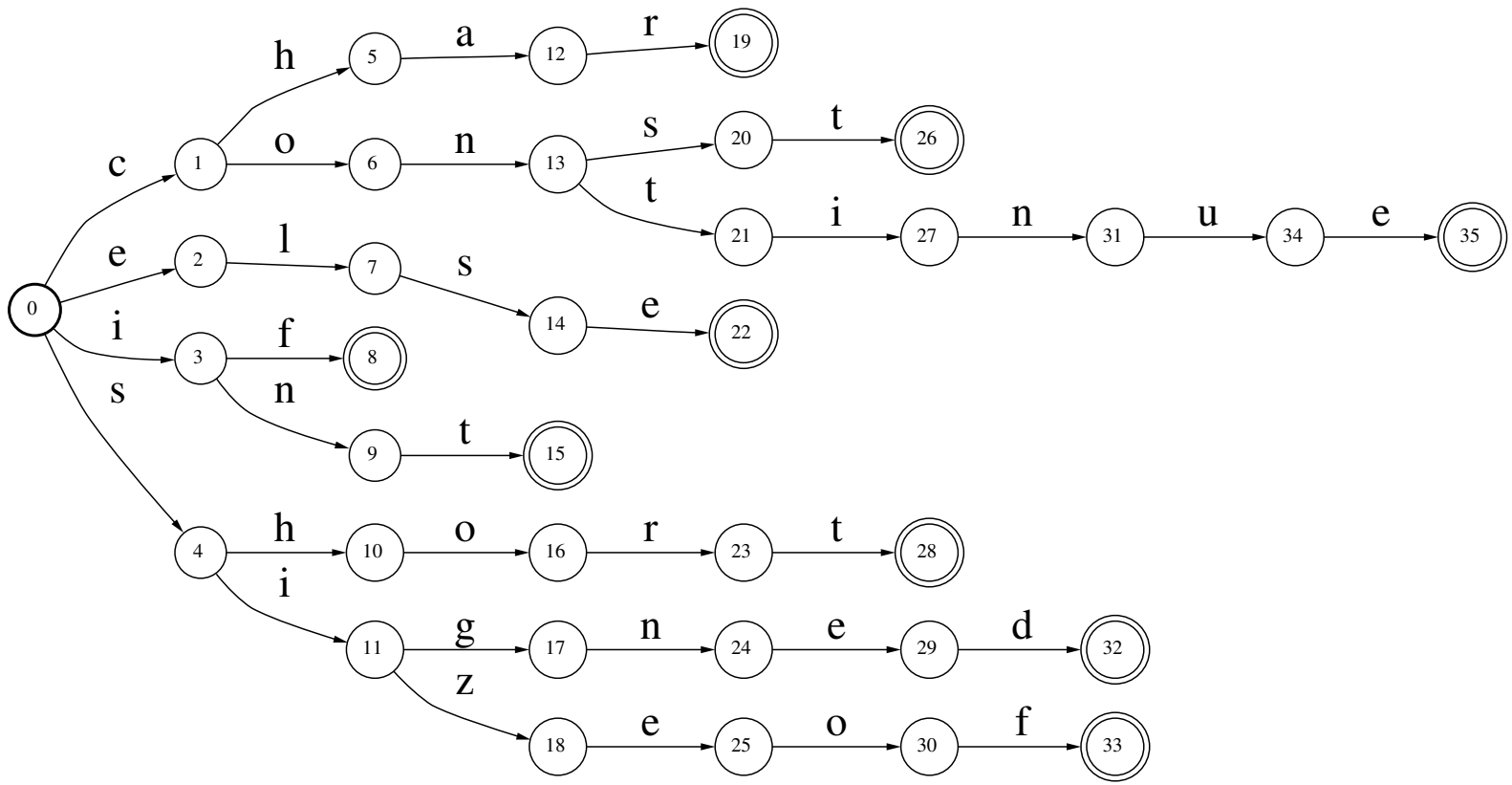


- `Intersect(token, keywords, &result);`

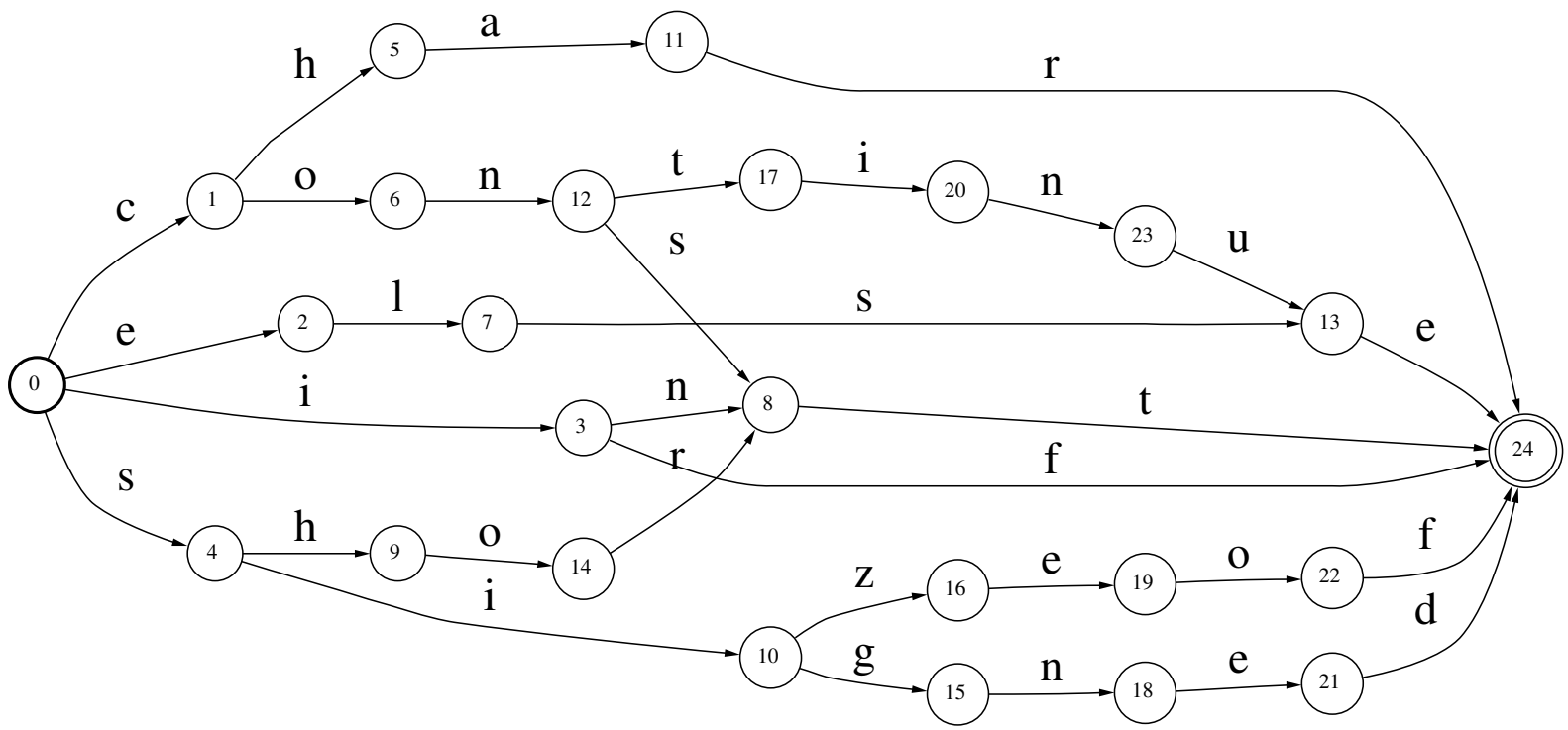
Keyword Detection – Automata Search (continued)



Keyword Detection – Deterministic Search

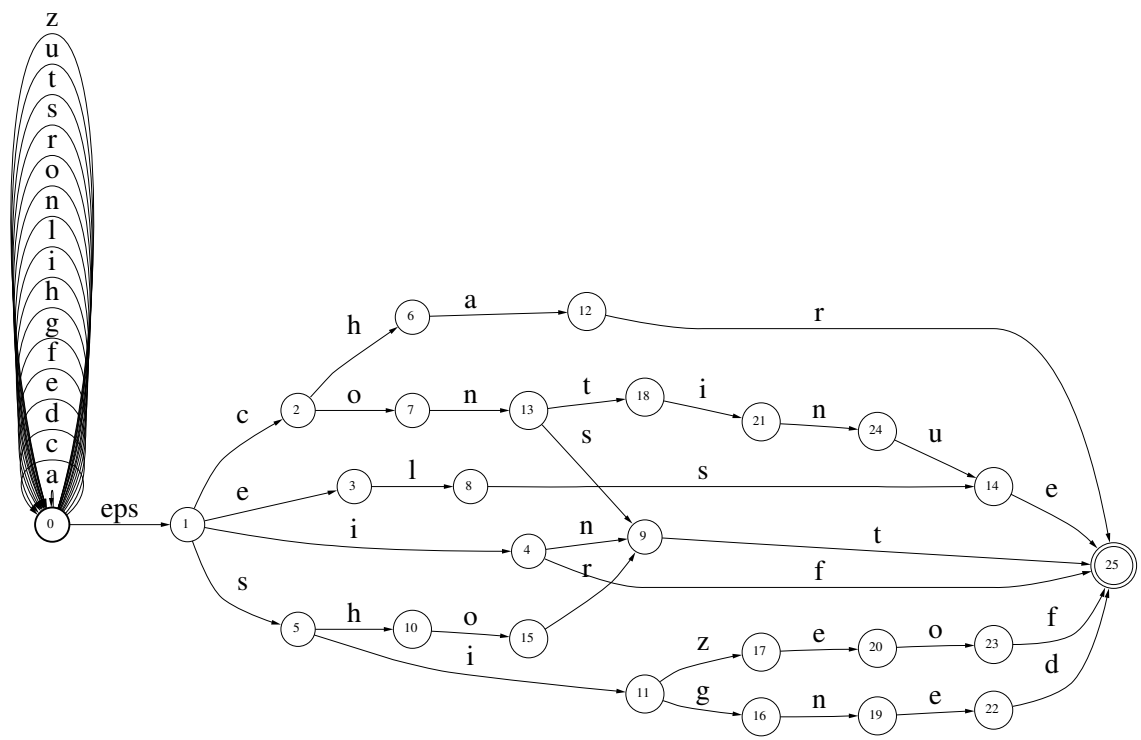


Keyword Detection – Minimal Deterministic Search



Pattern Matching – Dictionaries

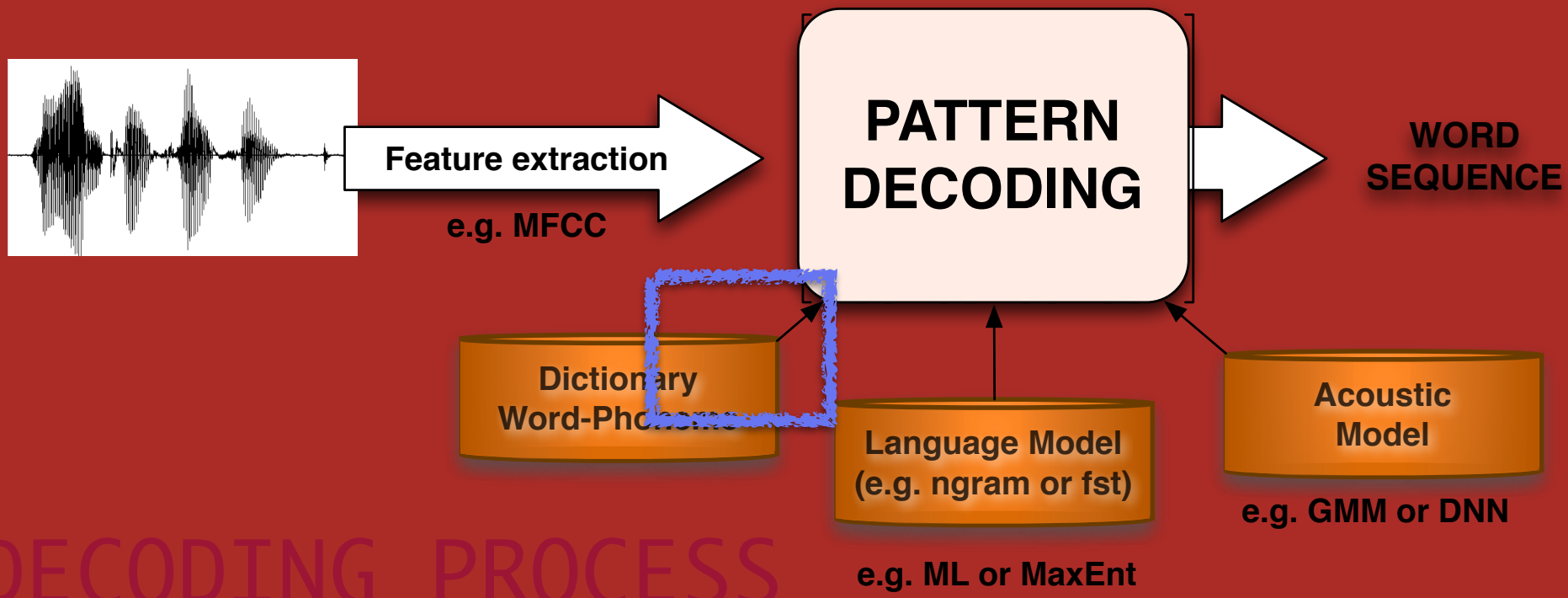
- Equation: $M = \Sigma^* D$
- Graphical Representation:



- Determinize using *failure transitions*; search by intersecting token string with automaton.

Example FST

- OpenFST example, Letters, Words



+ head -n 100 **letters.syms**

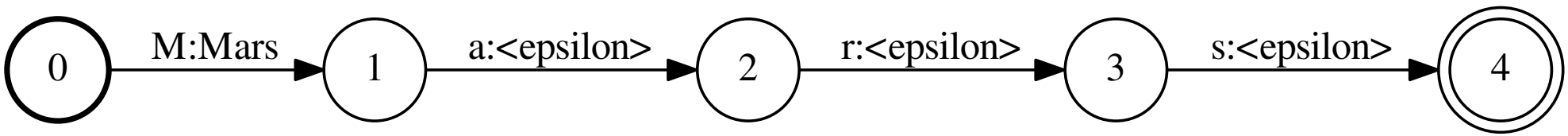
```
<epsilon>      0
<space> 32
!      33
"      34
#      35
$      36
%      37
&      38
'      39
(      40
)      41
*      42
+      43
,      44
-      45
.      46
/      47
0      48
1      49
2      50
3      51
4      52
5      53
6      54
7      55
8      56
9      57
:      58
.
.
.
.
```

+ head -n 100 **words.syms**

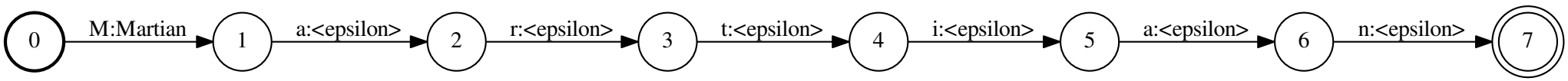
```
<epsilon>      0
No     1
one    2
would  3
have   4
believed       5
in     6
the    7
last   8
years  9
of     10
nineteenth     11
century 12
that   13
this   14
world  15
was    16
being  17
watched 18
keenly  19
and    20
closely 21
by     22
intelligences  23
greater 24
than   25
man    26
s      27
yet    28
.
.
.
```

```
fstcompile --isymbols=letters.syms --osymbols=words.syms >Mars.fst <<EOF
0 1 M Mars
1 2 a <epsilon>
2 3 r <epsilon>
3 4 s <epsilon>
4
EOF
```

```
fstdraw --isymbols=letters.syms --osymbols=words.syms -portrait Mars.fst | dot -Tpdf >Mars.pdf
```

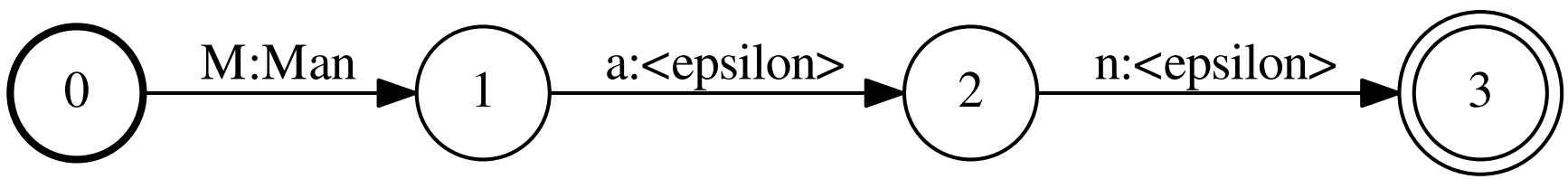


```
fstcompile --isymbols=letters.syms --osymbols=words.syms >Martian.fst <<EOF
0 1 M Martian
1 2 a <epsilon>
2 3 r <epsilon>
3 4 t <epsilon>
4 5 i <epsilon>
5 6 a <epsilon>
6 7 n <epsilon>
7
EOF
fstdraw --isymbols=letters.syms --osymbols=words.syms -portrait Martian.fst | dot -Tpdf
>Martian.pdf
```



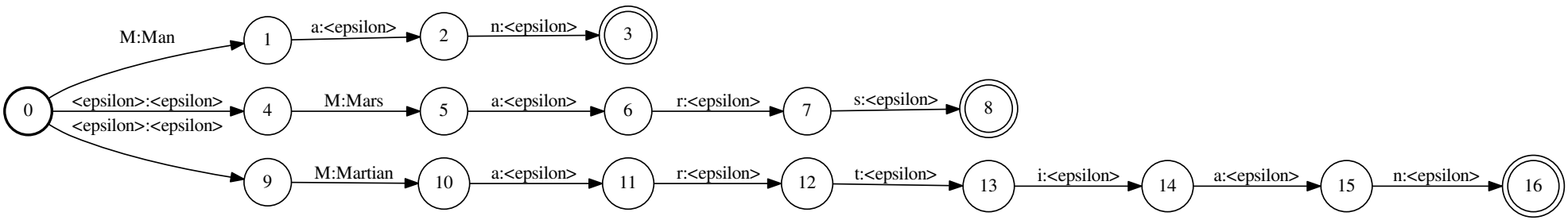
```
fstcompile --isymbols=letters.syms --osymbols=words.syms >Man.fst <<EOF
0 1 M Man
1 2 a <epsilon>
2 3 n <epsilon>
3
EOF
read
```

```
fstdraw --isymbols=letters.syms --osymbols=words.syms -portrait Man.fst | dot -Tpdf >Man.pdf
```

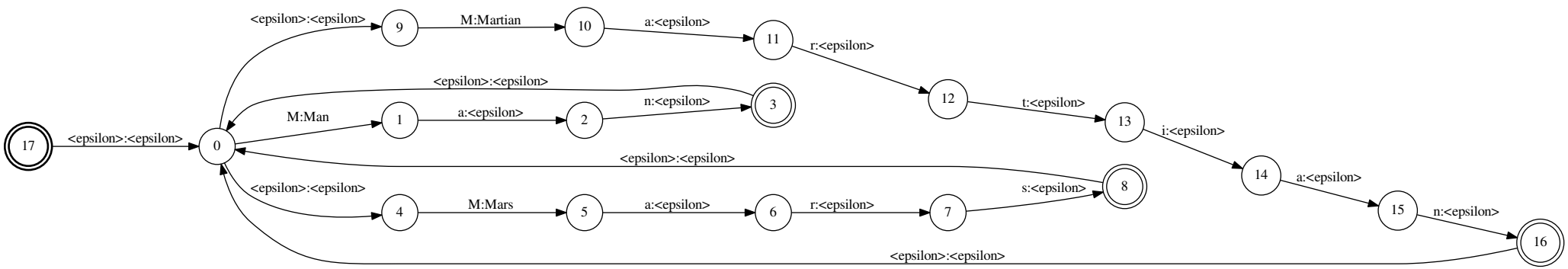


fstunion Man.fst Mars.fst | fstunion - Martian.fst >lexicon.fst

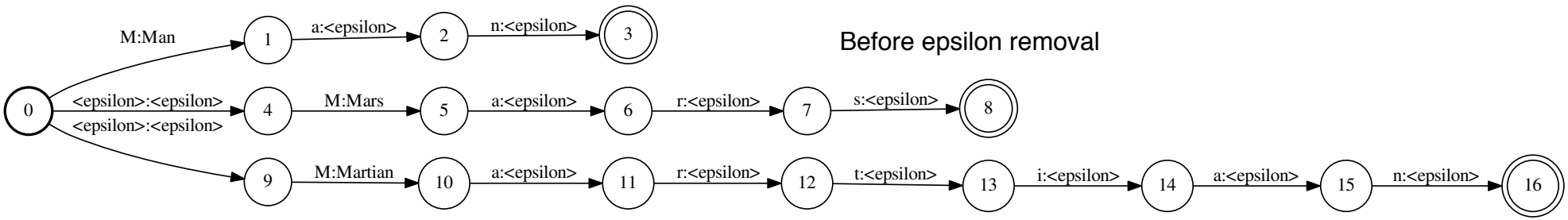
fstdraw --isymbols=letters.syms --osymbols=words.syms -portrait lexicon.fst | dot -Tpdf >lexicon.pdf

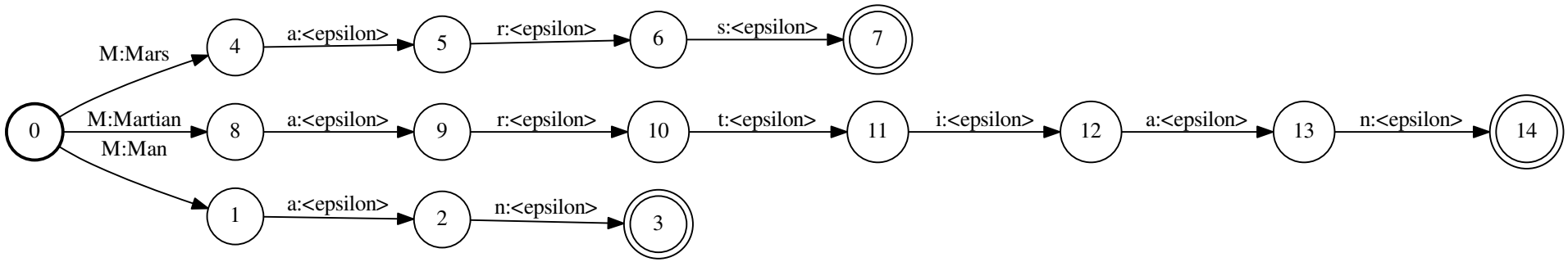


```
fstunion Man.fst Mars.fst | fstunion - Martian.fst |  fstclosure | \
  fstdraw --isymbols=letters.syms --osymbols=words.syms -portrait - | \
  dot -Tpdf >lexicon_closure.pdf
```

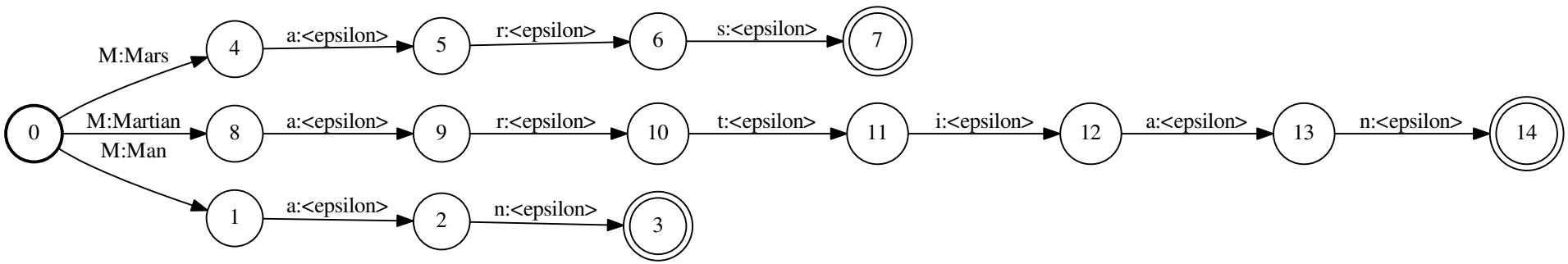


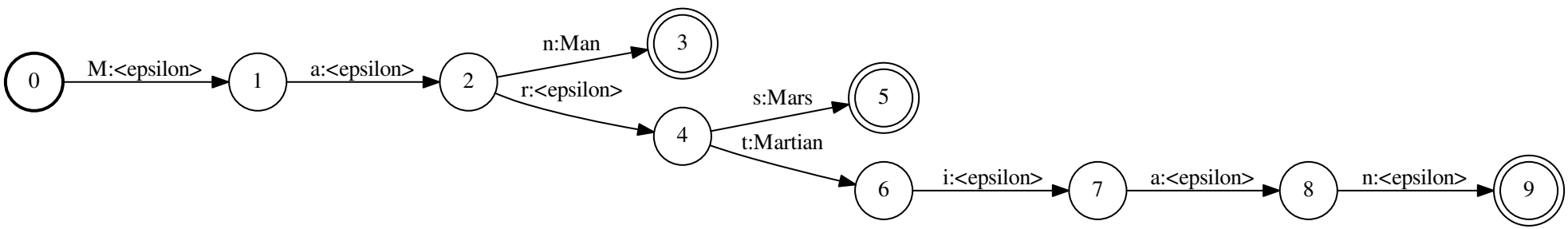
```
fstrmepsilon lexicon.fst | \
fstdraw --isymbols=letters.syms --osymbols=words.syms -portrait - | dot -Tpdf
>lexicon_rmepsilon.pdf
```





```
fstrmepsilon lexicon.fst | fstdeterminize  | \
fstdraw --isymbols=letters.syms --osymbols=words.syms -portrait - | \
dot -Tpdf >lexicon_rmepsilon_determinize.pdf
```

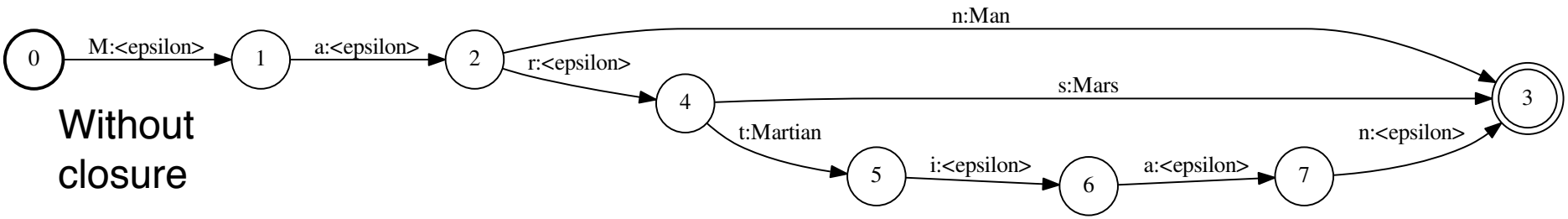




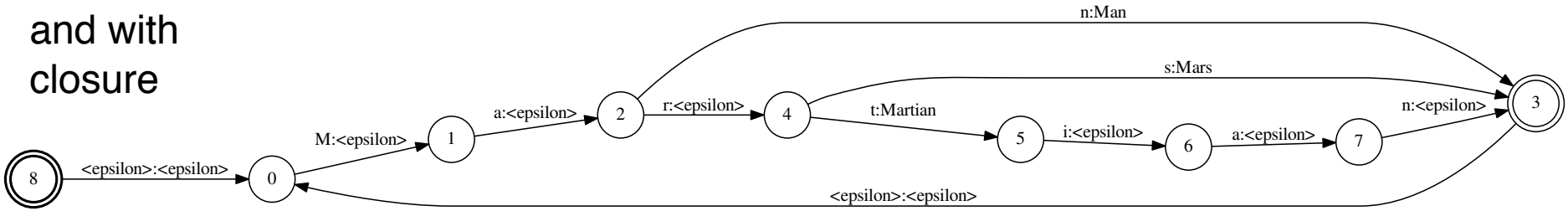
fstrmepsilon lexicon.fst | fstdeterminize | **fstminimize** |**fstclosure** > lexicon_opt.fst

fstdraw --isymbols=letters.syms --osymbols=words.syms -portrait lexicon_opt.fst | dot -Tpdf >lexicon_rmepsilon_determinize_minimize_closure.pdf

Without closure

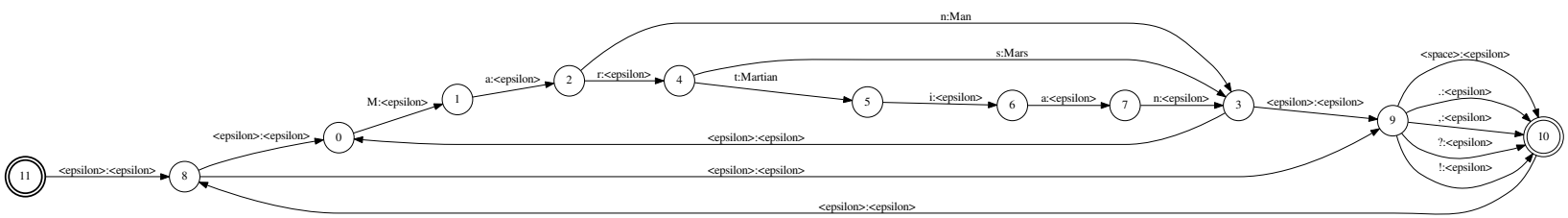


and with closure

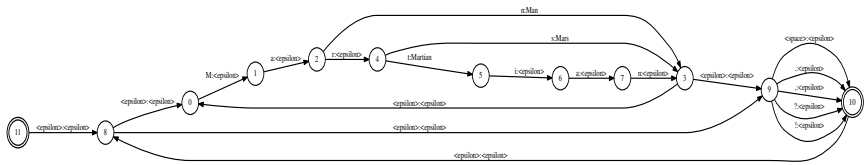


› Lets build a transducer that eats away punctuation from the input:

```
fstcompile --isymbols=letters.syms --osymbols=words.syms >punct.fst <<EOF
0 1 <space> <epsilon>
0 1 . <epsilon>
0 1 , <epsilon>
0 1 ? <epsilon>
0 1 ! <epsilon>
1
EOF
```

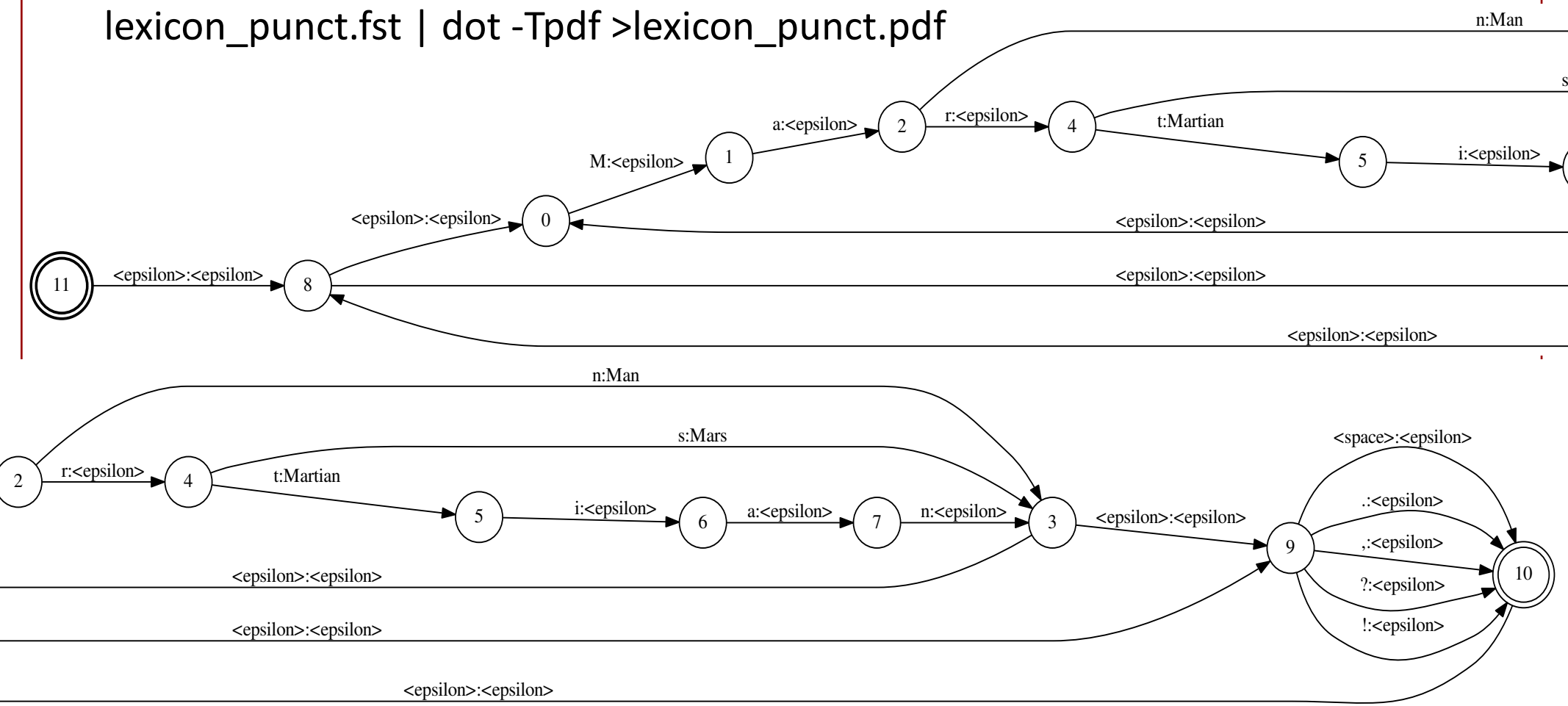


› And concatenate with the previous lexicon:



fstconcat lexicon_opt.fst punct.fst | fstclosure >lexicon_punct.fst

fstdraw --isymbols=letters.syms --osymbols=words.syms -portrait lexicon_punct.fst | dot -Tpdf >lexicon_punct.pdf



Lets build an fst for a 2 word sentence “Mars Man!”

fstcompile --isymbols=letters.syms --osymbols=letters.syms >MarsMan.fst <<EOF

0 1 M M

1 2 a a

2 3 r r

3 4 <epsilon> <epsilon>

4 5 s s

5 6 <space> <space>

6 7 M M

7 8 a a

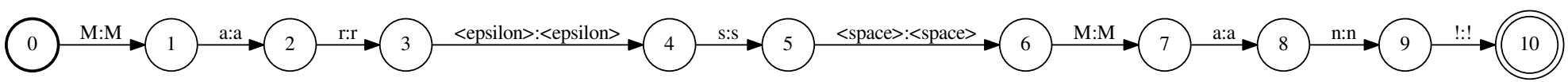
8 9 n n

9 10 ! !

10

EOF

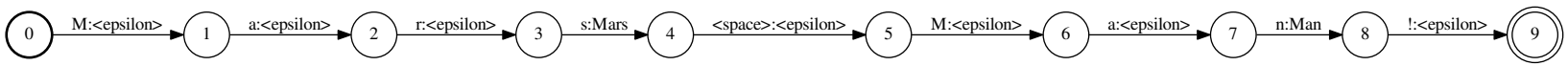
fstdraw --isymbols=letters.syms --osymbols=letters.syms -portrait MarsMan.fst | dot -Tpdf > MarsMan.pdf



We can use this to convert “Mars Man!” into clean text &|letters:

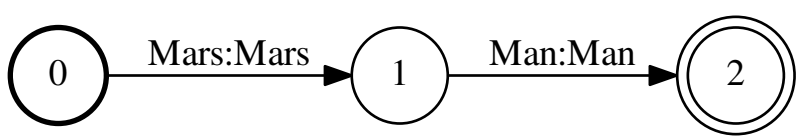
fstcompose MarsMan.fst lexicon_punct.fst | fstrmepsilon >tokens.fst

fstdraw --isymbols=letters.syms --osymbols=words.syms -portrait tokens.fst | dot -Tpdf > tokens.pdf



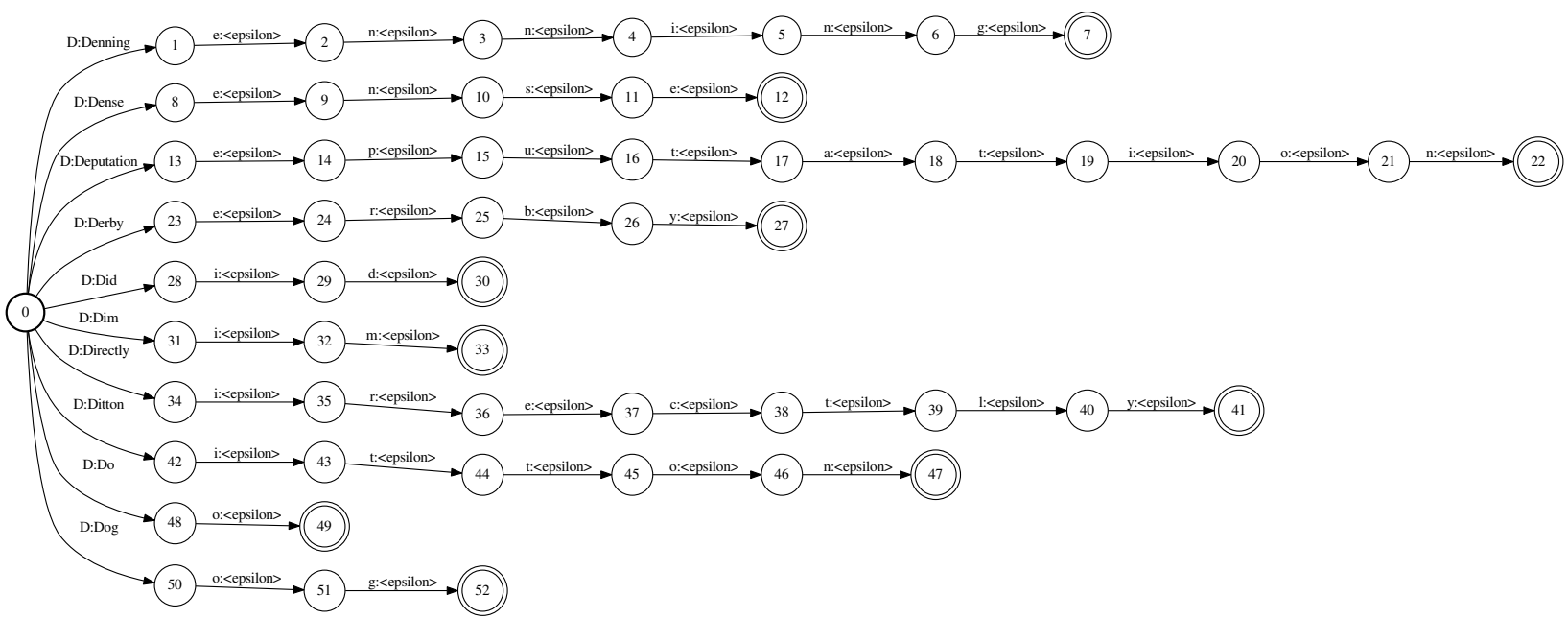
fstcompose MarsMan.fst lexicon_punct.fst | fstproject --project_output | fstrmepsilon >tokens.fst

fstdraw --isymbols=words.syms --osymbols=words.syms -portrait tokens.fst | dot -Tpdf > tokens_projected.pdf



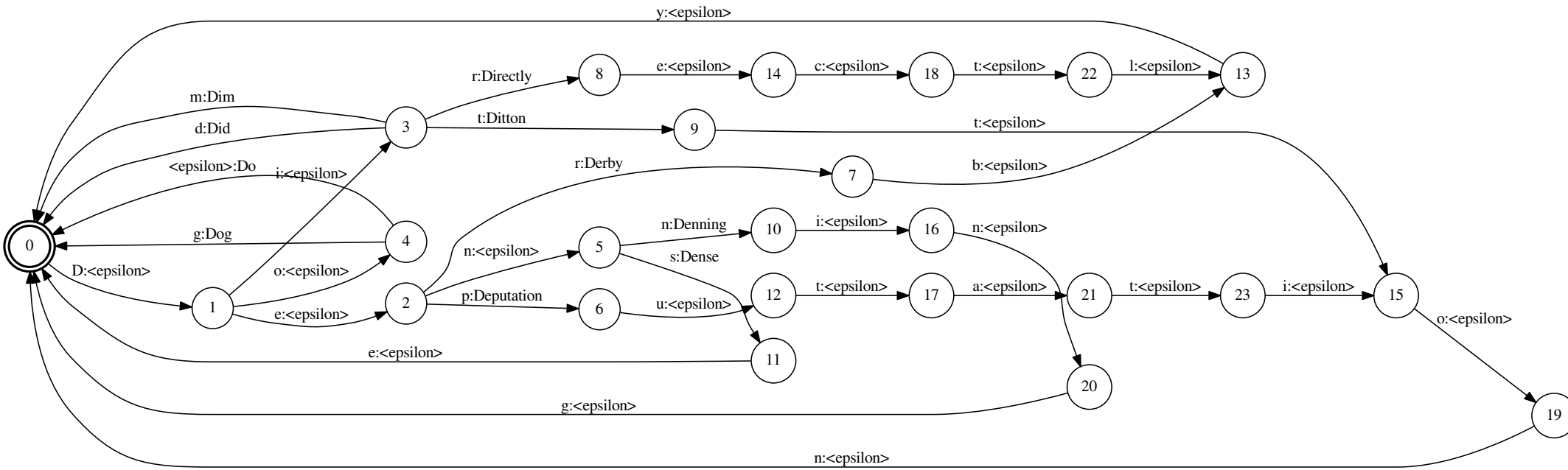
SLIDE TITLE

- cat words.syms | sort | sed '150,160!d' | python2.7 make_lex.py > some_words.txt
- fstcompile --isymbols=letters.syms --osymbols=words.syms some_words.txt > some_words_lexicon.fst
- fstdraw --isymbols=letters.syms --osymbols=words.syms -portrait some_words_lexicon.fst | dot -Tpdf > some_words_lexicon.pdf
- fstrmepsilon some_words_lexicon.fst | fstdeterminize | fstrmepsilon | fstminimize |fstclosure | fstrmepsilon | fstminimize > some_words_lexicon_opt.fst
- fstdraw --isymbols=letters.syms --osymbols=words.syms -portrait some_words_lexicon_opt.fst | dot -Tpdf > some_words_lexicon_opt.pdf
-



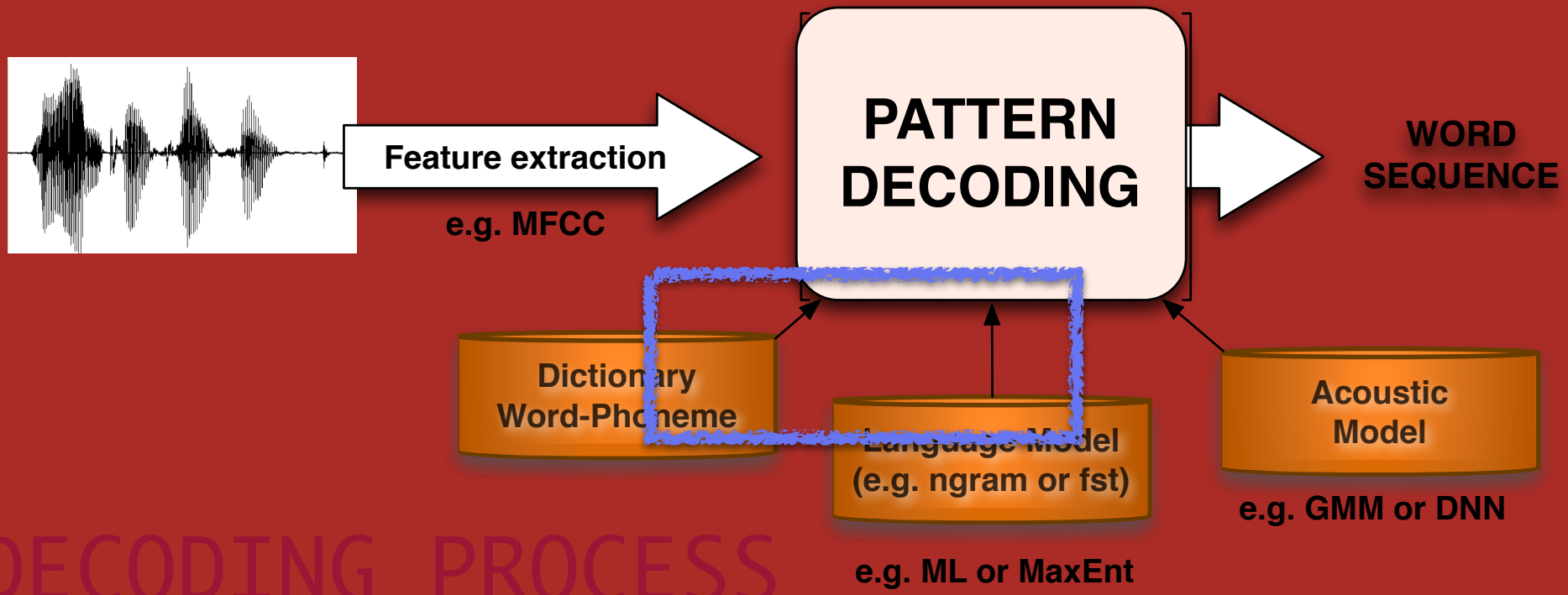
SLIDE TITLE

- cat words.syms | sort | sed '150,160!d' | python2.7 make_lex.py > some_words.txt
- fstcompile --isymbols=letters.syms --osymbols=words.syms some_words.txt > some_words_lexicon.fst
- fstdraw --isymbols=letters.syms --osymbols=words.syms -portrait some_words_lexicon.fst | dot -Tpdf > some_words_lexicon.pdf
- fstrmepsilon some_words_lexicon.fst | fstdeterminize | fstrmepsilon | fstminimize |fstclosure | fstrmepsilon | fstminimize > some_words_lexicon_opt.fst
- fstdraw --isymbols=letters.syms --osymbols=words.syms -portrait some_words_lexicon_opt.fst | dot -Tpdf > some_words_lexicon_opt.pdf
-



The tiniest dictionary & language model

- Closer to what kaldi does; but super tiny toy example



DECODING PROCESS

DICTIONARY

cat dict

two T UW
to T UW
to T IH
to T AH
too T UW

This could have alternate pron symbols as well, e.g.

two T UW
to T UW
to(2) T IH
to(3) T AH
too T UW

cat text

to two

two to

two to

two to

too

This just shows that “two to” appears a lot more frequently than “to two” or “too”.

SYMBOLS LIST

- We need a list of word symbols:

cat words.syms

<eps> 0
to 1
two 2
too 3
#0 4
#1 5
#2 6
#3 7
#4 8

This also has a bunch of disambiguation symbols #…
Necessary as we will see later.

BUILD LANGUAGE MODELS

› We can easily build language models with SRILM:

```
ngram-count -order 1 -text text -lm lm.1.arpa

cat lm.1.arpa

\data\
ngram 1=5

\1-grams:
-0.447158      </s>
-99    <s>
-0.544068      to
-1.146128      too
-0.544068      two

\end\
```

2-GRAMS (BI-GRAMS)


```
ngram-count -order 2 -text text -lm lm.2.arpa
cat lm.2.arpa

\data\
ngram 1=5
ngram 2=8

\1-grams:
-0.447158      </s>
-99    <s>    -0.7569618
-0.544068      to     -0.4559319
-1.146128      too    -0.4101745
-0.544068      two    -0.4559319

\2-grams:
-0.5509075     <s> to
-0.5509075     <s> too
-0.4259687     <s> two
-0.30103       to </s>
-0.4259687     to two
-0.1249387     too </s>
-0.4259687     two </s>
-0.30103       two to

\end\
```

3-GRAMS (TRIGRAMS)


```
ngram-count -order 3 -text text -lm lm.3.arpa
cat lm.3.arpa

\data\
ngram 1=5
ngram 2=8
ngram 3=2

\1-grams:
-0.447158      </s>
-99    <s>    -0.7569618
-0.544068      to     -0.4559319
-1.146128      too    -0.4101745
-0.544068      two    -0.4559319

\2-grams:
-0.5509075     <s> to
-0.5509075     <s> too
-0.4259687     <s> two -0.30103
-0.30103       to </s>
-0.4259687     to two
-0.1249387     too </s>
-0.4259687     two </s>
-0.30103       two to  -0.30103

\3-grams:
-0.1249387     two to </s>
-0.1249387     <s> two to

\end\
```

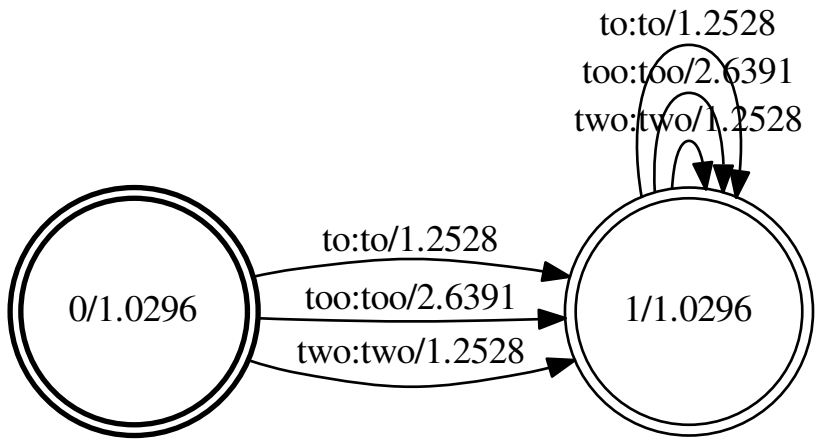
ARPA computation:

$$P(w_N \mid w_{N-1}, w_{N-2}, \ldots, w_1) = P(w_N \mid w_{N-1}, w_{N-2}, \ldots, w_2) * \text{backoff-weight}(w_{N-1} \mid w_{N-2}, \ldots, w_1)$$

LANGUAGE MODEL AS FST


```
cat lm.1.arpa | \
    grep -v '<s> <s>' | \
    grep -v '</s> <s>' | \
    grep -v '</s> </s>' | \
    arpa2fst - | \
    fstprint | \
    ./eps2disambig.pl |\
    ./s2eps.pl | \
    fstcompile --isymbols=words.syms \
      --osymbols=words.syms  \
      --keep_isymbols=false --keep_osymbols=false | \
    fstrmepsilon > G.1.fst

fstdraw  --isymbols=words.syms  --osymbols=words.syms  -portrait G.1.fst | dot -Tpdf
> G.1.pdf
```

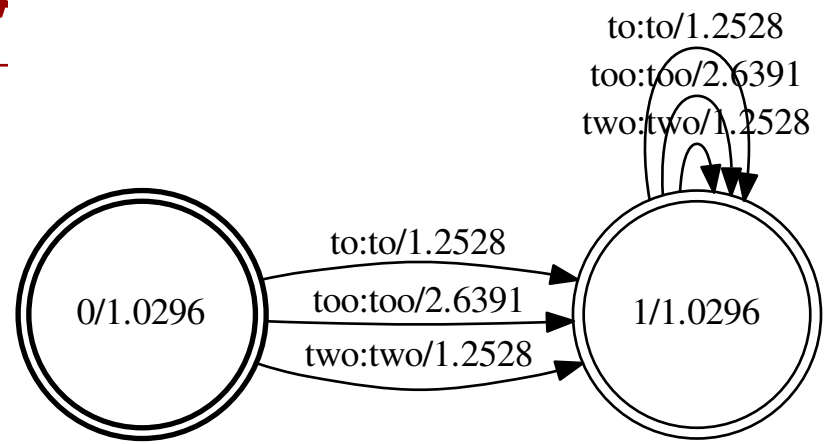
LANGUAGE MODEL A

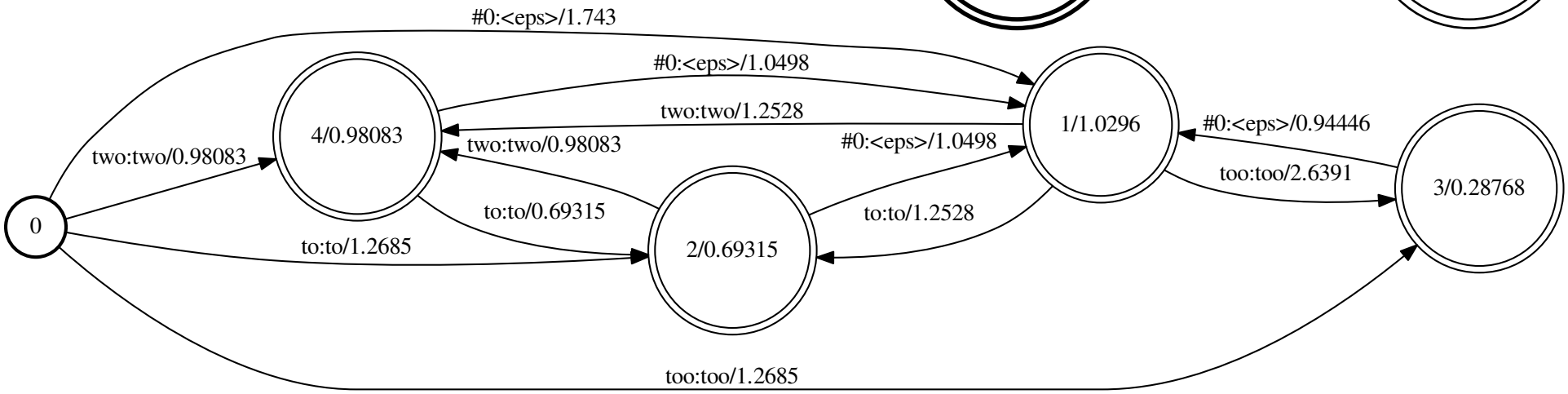



```
cat lm.1.arpa | \
    grep -v '<s> <s>' | \
    grep -v '</s> <s>' | \
    grep -v '</s> </s>' | \
    arpa2fst - | \
    fstprint | \
    ./eps2disambig.pl |\
    ./s2eps.pl | \
    fstcompile --isymbols=words.syms \
      --osymbols=words.syms  \
      --keep_isymbols=false --keep_osymbols=false | \
    fstrmepsilon > G.1.fst
```

```
fstdraw  --isymbols=words.syms  --osymbols=words.syms  -portrait G.1.fst | dot -Tpdf
> G.1.pdf
```

LANGUAGE MODEL A

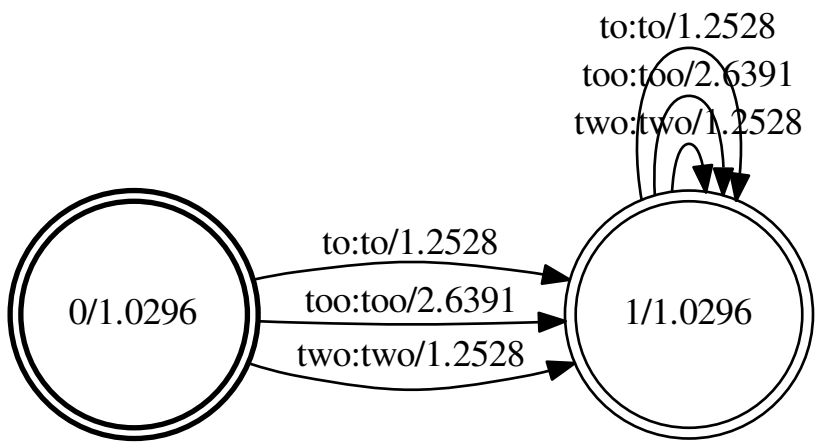


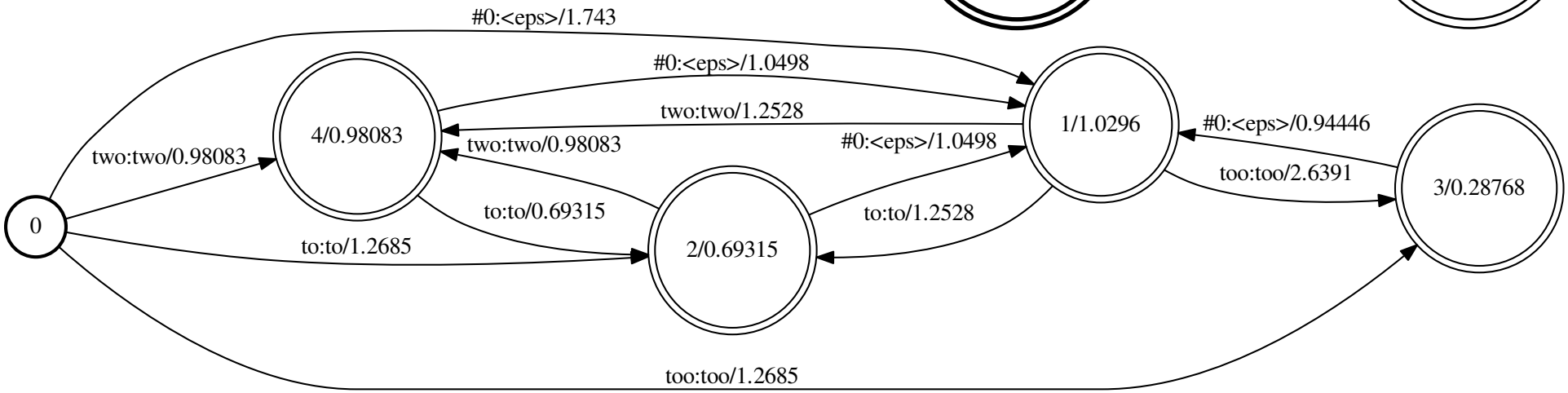


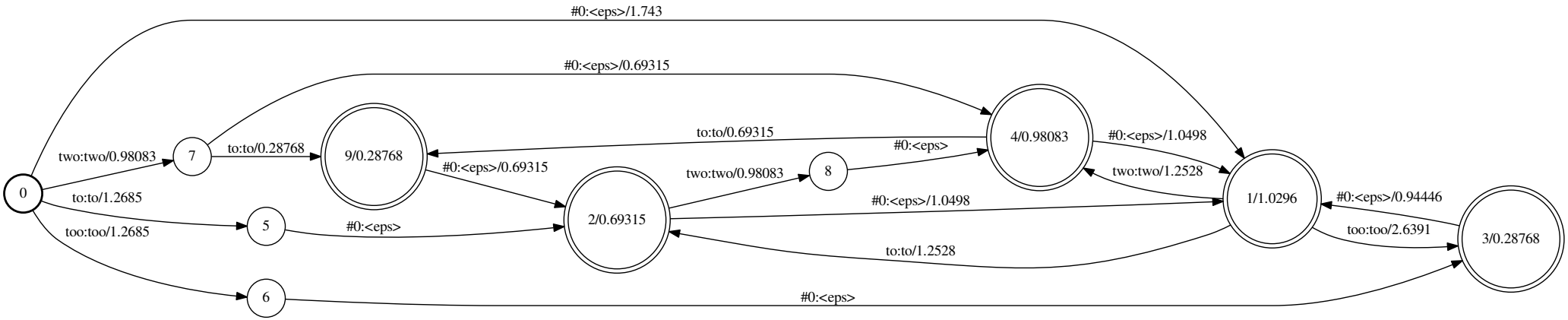

```
› cat lm.1.arpa | \
›     grep -v '<s> <s>' | \
›     grep -v '</s> <s>' | \
›     grep -v '</s> </s>' | \
›
›
›
›
›
›
›
›       --keep_isymbols=false --keep_osymbols=false | \
›     fstrmepsilon > G.1.fst

› fstdraw  --isymbols=words.syms  --osymbols=words.syms  -portrait G.1.fst | dot -Tpdf
  > G.1.pdf
›
```

LANGUAGE MODEL A

```
cat lm.1.arpa | \
    grep -v '<s> <s>' | \
    grep -v '</s> <s>' | \
    grep -v '</s> </s>' | \
    --keep_isymbols=false --keep_osymbols=false | \
```







PHONES -> WORDS

Now we want to transition on our fst consuming phonemes and outputting words.

So need symbols for phonemes:

```
cat phones.syms
<eps> 0
sil 1
T 2
UW 3
IH 4
AH 5
#0 6
#1 7
#2 8
#3 9
#4 10
```

and also need to disambiguate same-pronunciation words:

```
cat dict_disambig
two    T UW #1
to     T UW #2
to     T IH
to     T AH
too    T UW #3
```

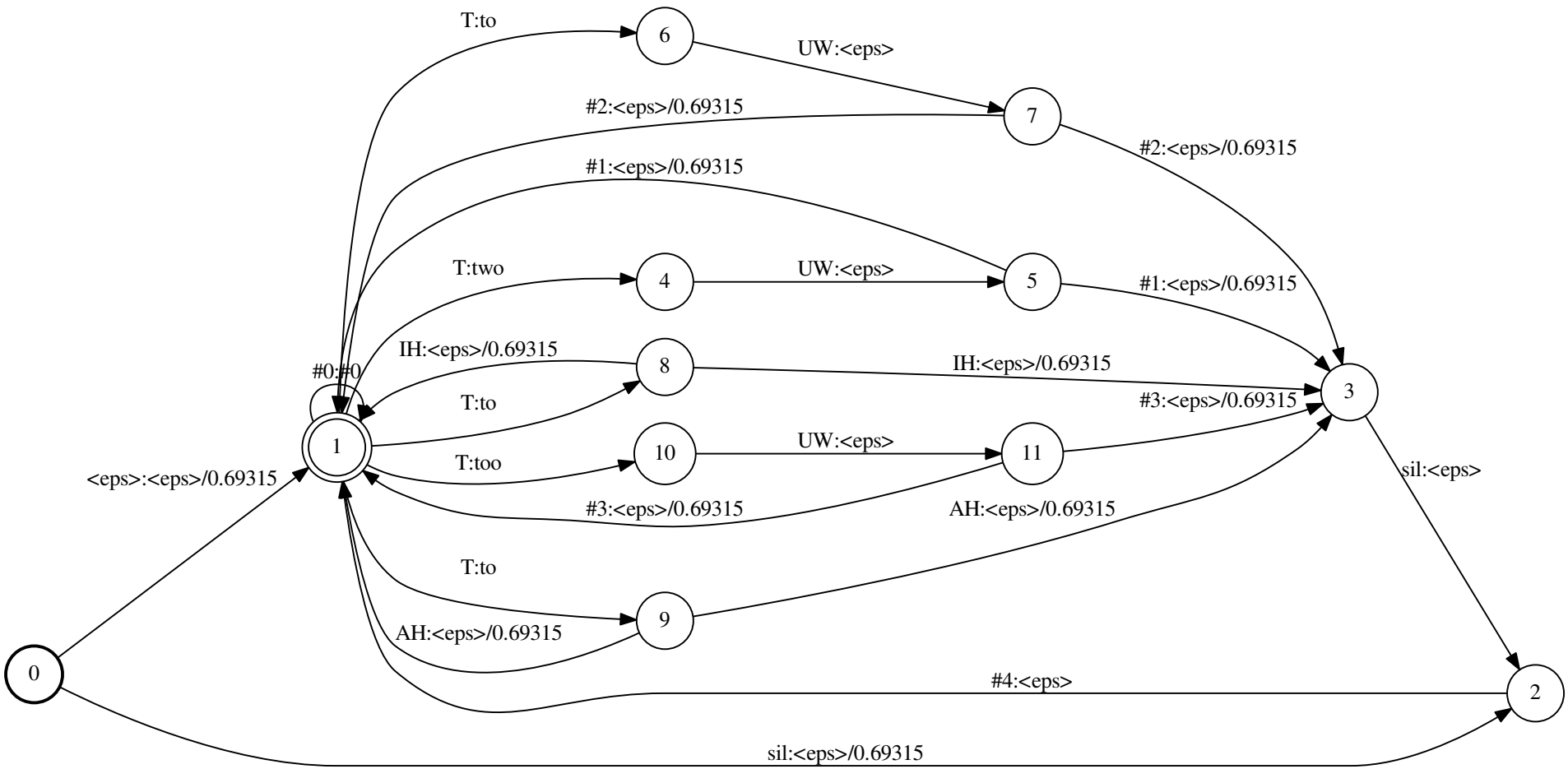
DISAMBIGUATING LEXICON


```
./make_lexicon_fst.pl dict_disambig 0.5 sil '#4' | \
  fstcompile --isymbols=phones.syms \
   --osymbols=words.syms \
   --keep_isymbols=false --keep_osymbols=false |\
  fstaddselfloops  phones.disambig words.disambig  | \
  fstarcsort --sort_type=olabel \
  > L_disambig.fst
```

```
fstdraw  --isymbols=phones.syms  --osymbols=words.syms  -portrait
L_disambig.fst | dot -Tpdf > L_disambig.pdf
```

DISAMBIGUATING LEXICON


```
./make_lexicon_fst.pl dict_disambig 0.5 sil '#4' | \
  fstcompile --isymbols=phones.syms \
   --osymbols=words.syms \
```



DISAMBIGUATING LEXICON

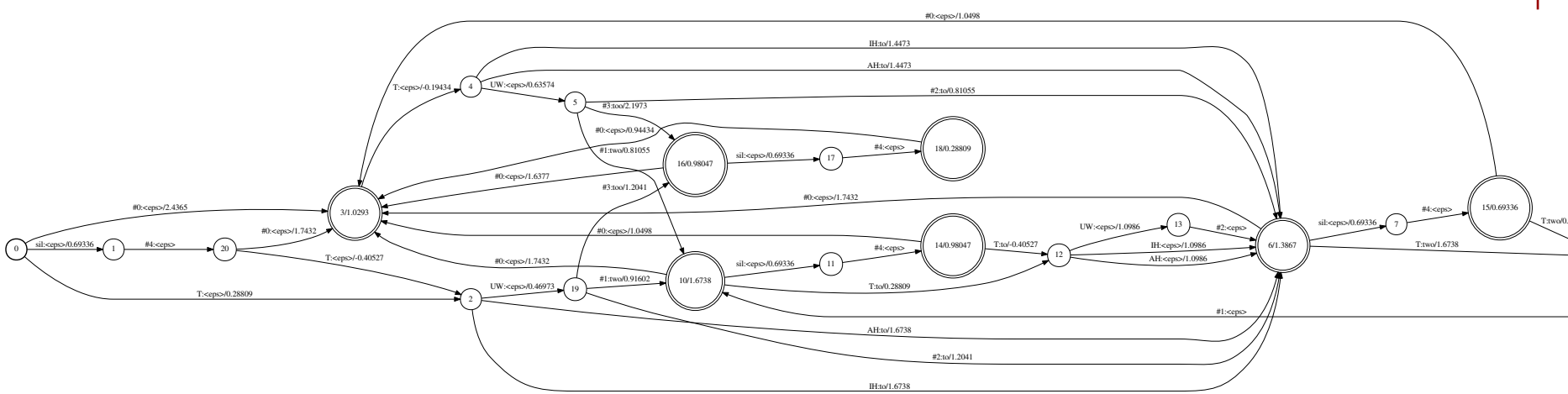

```
./make_lexicon_fst.pl dict_disambig 0.5 sil '#4' | \
  fstcompile --isymbols=phones.syms \
   --osymbols=words.syms \
   --keep_isymbols=false --keep_osymbols=false |\
  fstaddselfloops  phones.disambig words.disambig  | \
  fstarcsort --sort_type=olabel \
  > L_disambig.fst
```

```
fstdraw  --isymbols=phones.syms  --osymbols=words.syms  -portrait
L_disambig.fst | dot -Tpdf > L_disambig.pdf
```

COMPOSE LEXICON AND GRAMMAR


```
fsttablecompose L_disambig.fst G.fst |\
  fstdeterminizestar --use-log=true | \
  fstminimizeencoded  > LG.fst
```

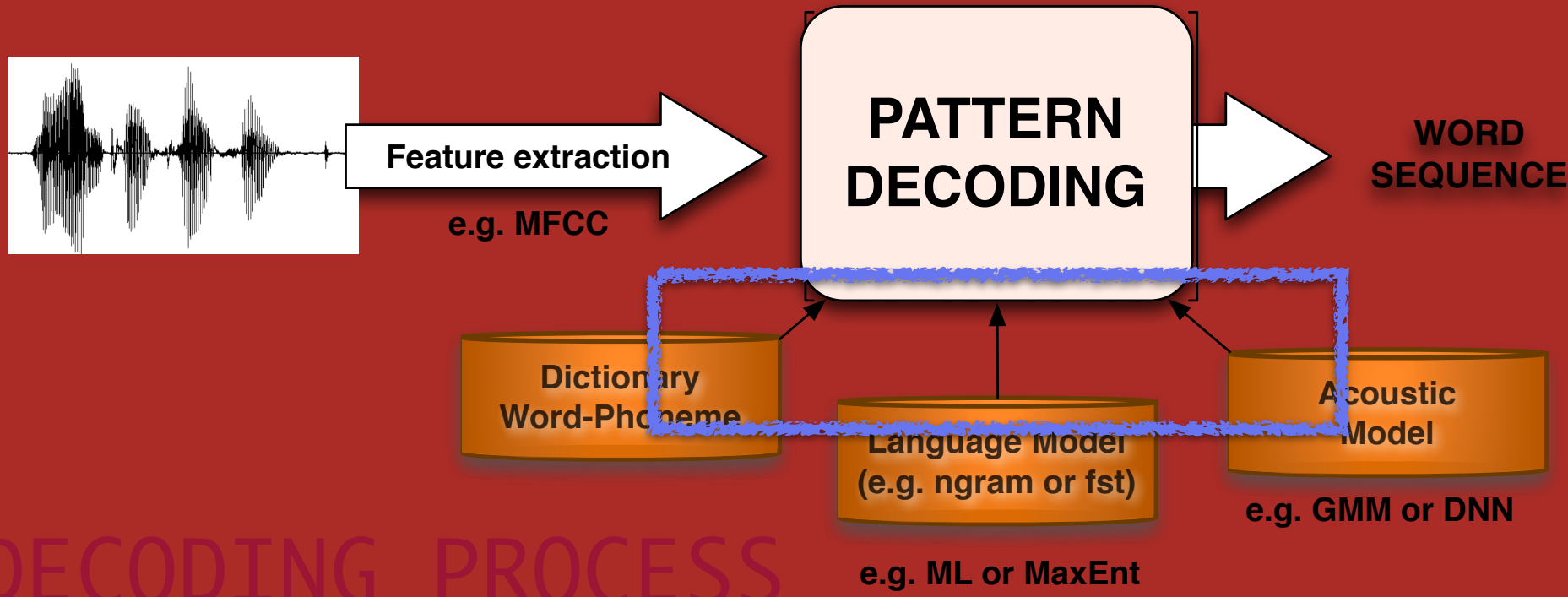
```
fstdraw  --isymbols=phones.syms  --osymbols=words.syms  -portrait LG.fst |
dot -Tpdf > LG.pdf
```



HCLG IN SHORT

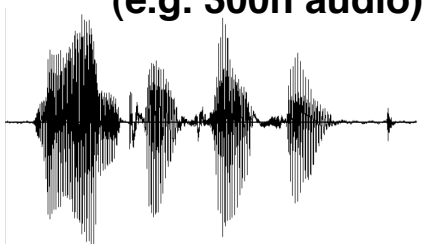
- I don't want to make it more confusing, but bottom line:
- G:
  - We saw this: It's the language model
- L:
  - We saw this: It's the lexicon
- C:
  - This is the context; often, (and in s5) this is a triphone context
  - It will map before/center/after:center [sometimes kaldi has these as b/a/c]
  - It will also (at composition time) map unseen triphones to a seen one.
    e.g.
    Since our silence phoneme is context independent all
    /sil/:ah:sil:ah (where ah:sil:ah can be just a random chosen triphone… all are collapsed to one in pdfs)
    or:
    "<eps>/ey/<eps>:ey/ey/<eps>" says that these two share the same pdf's as well
- H:
  - Represents the mapping to pdf ids.
  - e.g. P_2_1234_1
    Phoneme P,
    3rd state (counting from zero)
    is captured by pdfid 1234
    and we can leave this state from outgoing arc with ID 1
  - When the H links to DNN, 1234 refers to the 1234 output neuron of the DNN

HCLG



DECODING PROCESS

ASR Training Process



(e.g. 300h audio)

Separate speech from background first

Feature extraction

feature vector | feature vector | feature vector | feature vector | feature vector

Transcript of above

Dictionary Word-Phoneme

Phonetic transcription

Mostly human-made, especially in non-phonetic languages like English

Training (e.g. Baum Welch)

Models

TRAINING PROCESS

Gaussian Mixture Acoustic Model

Millions of words of representative transcripts for the domain

Feature extraction

Language Model (e.g. ngram)

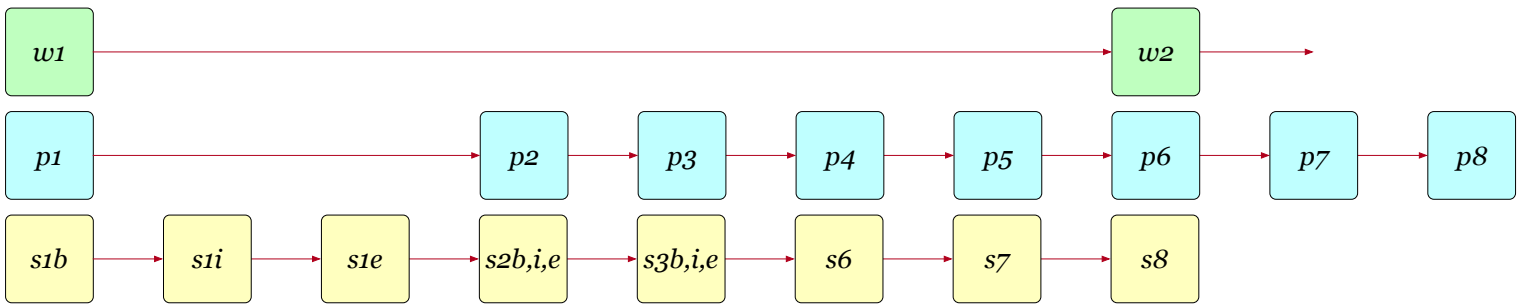
ASR MODEL

An utterance has

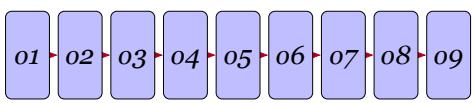
Multiple words with $P(w_i|w_{i-1},w_{i-2})$...

Each composed of multiple phonemes

Each phoneme composed of 3 states



And each CD state can be activated by one or more observations



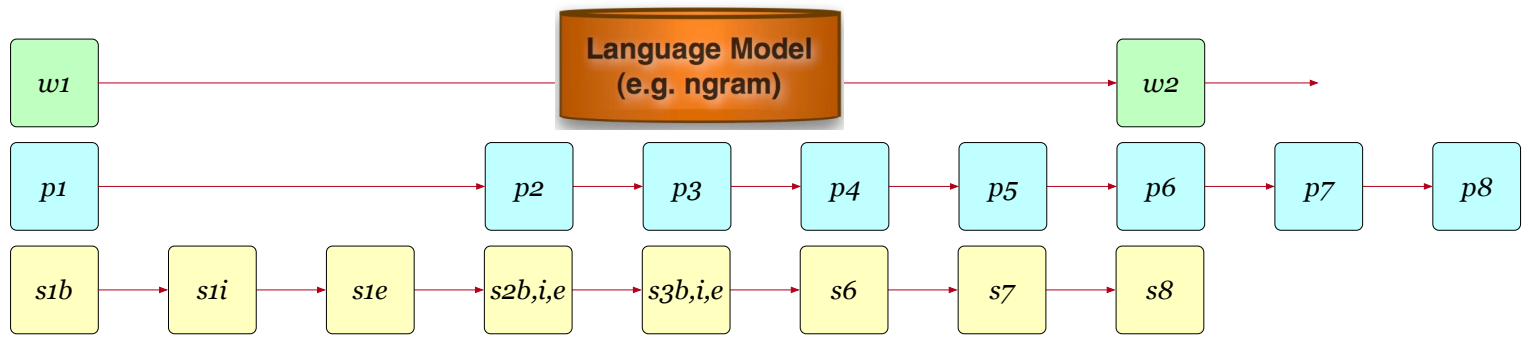
ASR MODEL

An utterance has

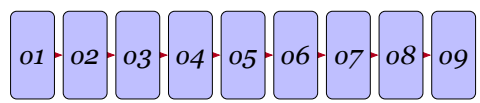
Multiple words with $P(w_i|w_{i-1},w_{i-2})$...

Each composed of multiple phonemes

Each phoneme composed of 3 states



And each CD state can be activated by one or more observations



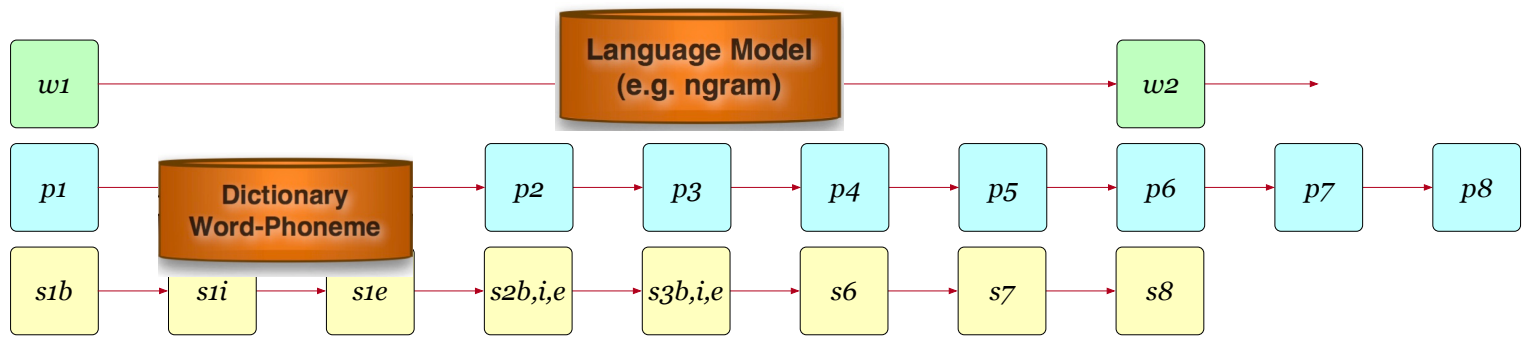
ASR MODEL

An utterance has

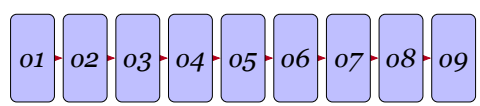
Multiple words with $P(w_i|w_{i-1},w_{i-2})$...

Each composed of multiple phonemes

Each phoneme composed of 3 states



And each CD state can be activated by one or more observations



ASR MODEL

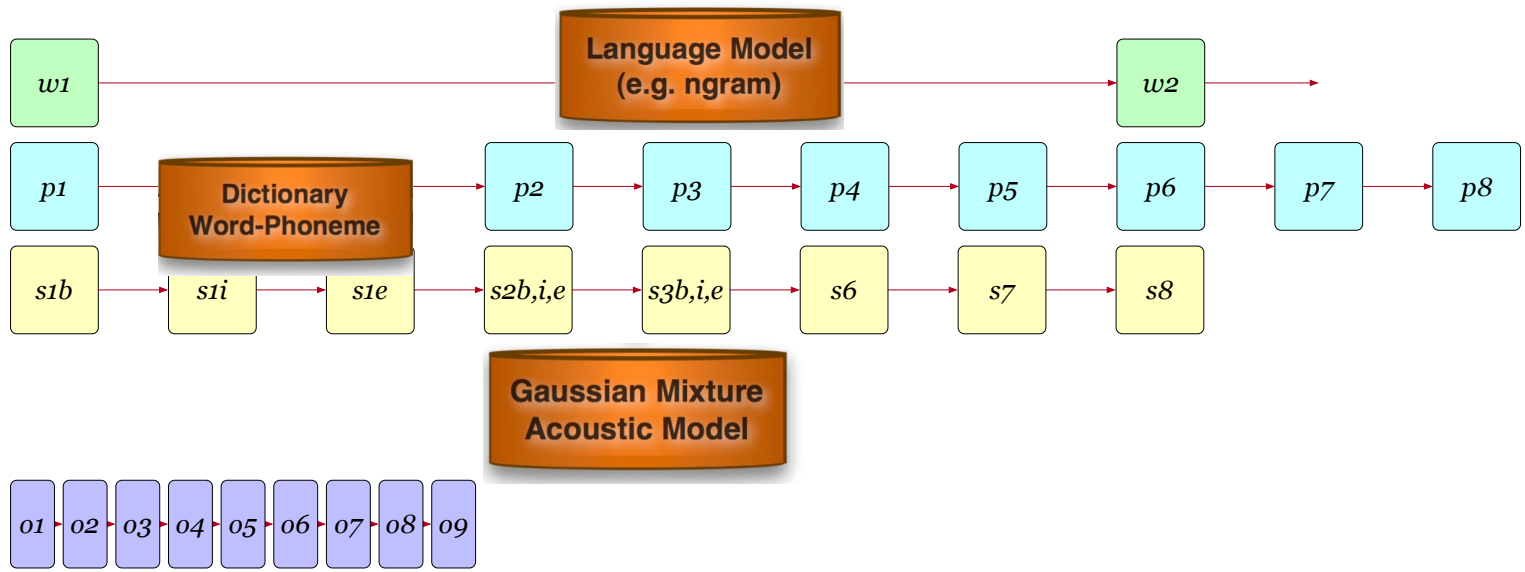
An utterance has

Multiple words with $P(w_i|w_{i-1},w_{i-2})$...

Each composed of multiple phonemes

Each phoneme composed of 3 states

And each CD state can be activated by one or more observations



ASR MODEL

An utterance has

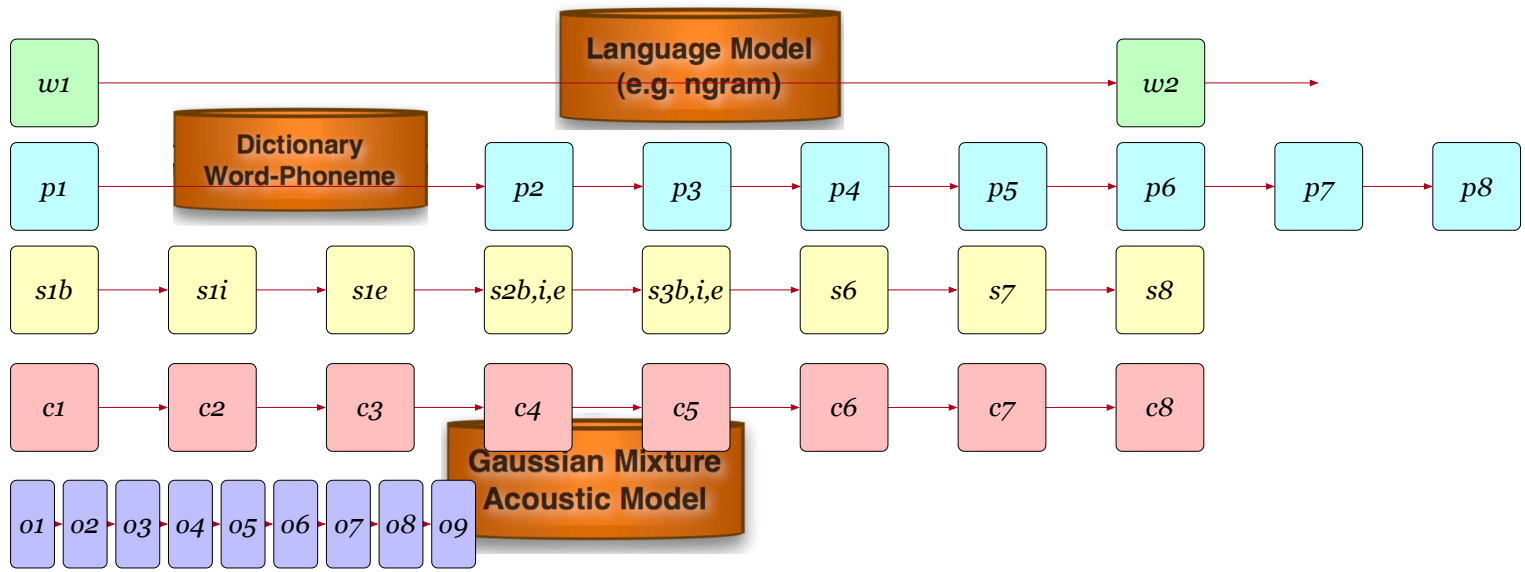
Multiple words with $P(w_i|w_{i-1},w_{i-2})$...

Each composed of multiple phonemes

Each phoneme composed of 3 states

Each state represented by a CD state model (context dependent tied state, pdfid)

And each CD state can be activated by one or more observations



ASR MODEL

- H=HMM or DNN model
- C=Context Dependent model
- L=Language/Lexicon information
- G=Grammar, Language model