LSA 352:
Speech Recognition and Synthesis

Dan Jurafsky

Lecture 5: Intro to ASR+HMMs: Forward, Viterbi, Baum-Welch

IP Notice:

Outline for Today

- Speech Recognition Architectural Overview
- Hidden Markov Models in general
  - Forward
  - Viterbi Decoding
  - Baum-Wlech
- Applying HMMs to speech
- How this fits into the ASR component of course
  - July 6: Language Modeling
  - July 19 (today): HMMs, Forward, Viterbi, Start of Baum-Welch (EM) training
  - July 23: Feature Extraction, MFCCs, and Gaussian Acoustic modeling
  - July 26: Evaluation, Decoding, Advanced Topics

LVCSR

- Large Vocabulary Continuous Speech Recognition
- ~20,000-64,000 words
- Speaker independent (vs. speaker-dependent)
- Continuous speech (vs isolated-word)

Current error rates

Ballpark numbers; exact numbers depend very much on the specific corpus

Task	Vocabulary	Error Rate%
Digits	11	0.5
WSJ read speech	5K	3
WSJ read speech	20K	3
Broadcast news	64,000+	10
Conversational Telephone	64,000+	20

HSR versus ASR

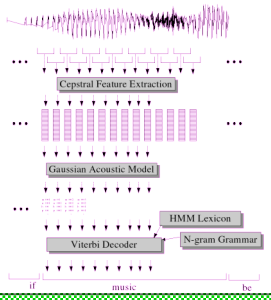
Task	**Vocab**	**ASR**	**Hum SR**
Continuous digits	**11**	**.5**	**.009**
WSJ 1995 clean	**5K**	**3**	**0.9**
WSJ 1995 w/noise	**5K**	**9**	**1.1**
SWBD 2004	**65K**	**20**	**4**

- Conclusions:
  - Machines about 5 times worse than humans
  - Gap increases with noisy speech
  - These numbers are rough, take with grain of salt

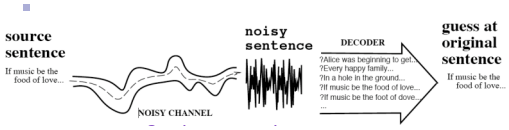
LVCSR Design Intuition

- Build a statistical model of the speech-to-words process
- Collect lots and lots of speech, and transcribe all the words.
- Train the model on the labeled speech
- Paradigm: Supervised Machine Learning + Search

Speech Recognition Architecture



The Noisy Channel Model



- Pick the one that is most probable given the waveform.

The Noisy Channel Model (II)

- What is the most likely sentence out of all sentences in the language L given some acoustic input O?
- Treat acoustic input O as sequence of individual observations
  - O = $o_1,o_2,o_3,\ldots,o_t$
- Define a sentence as a sequence of words:
  - W = $w_1,w_2,w_3,\ldots,w_n$

Noisy Channel Model (III)

- Probabilistic implication: Pick the highest prob S:

$$\hat{W} = \arg\max_{W \in L} P(W \mid O)$$

- We can use Bayes rule to rewrite this:

$$\hat{W} = \arg\max_{W \in L} \frac{P(O \mid W)P(W)}{P(O)}$$

- Since denominator is the same for each candidate sentence W, we can ignore it for the argmax:

$$\hat{W} = \arg\max_{W \in L} P(O \mid W)P(W)$$

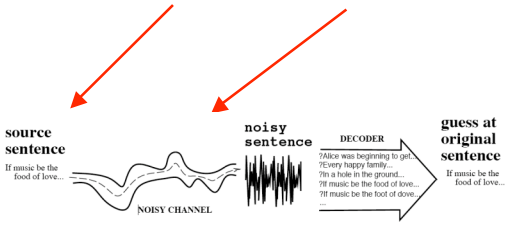
Noisy channel model

likelihood prior

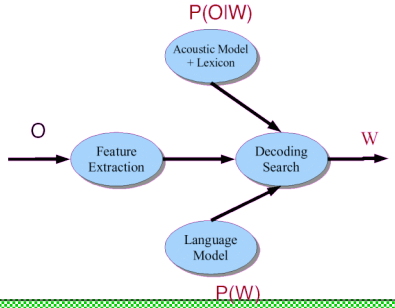
$$\hat{W} = \arg\max_{W \in L} P(O \mid W)P(W)$$

The noisy channel model

- Ignoring the denominator leaves us with two factors: P(Source) and P(Signal|Source)



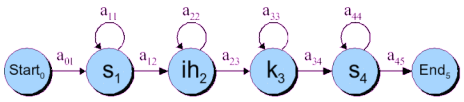
Speech Architecture meets Noisy Channel



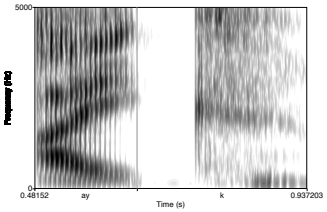
Architecture: Five easy pieces (only 2 for today)

- Feature extraction
- Acoustic Modeling
- HMMs, Lexicons, and Pronunciation
- Decoding
- Language Modeling

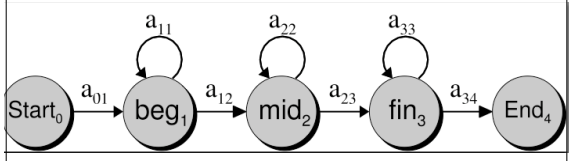
HMMs for speech



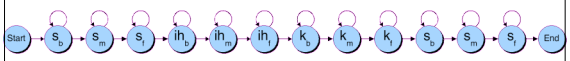
Phones are not homogeneous!



Each phone has 3 subphones



Resulting HMM word model for "six"



HMMs more formally

- Markov chains
- A kind of weighted finite-state automaton

$Q = q_1 q_2 \ldots q_N$ — a set of **states**

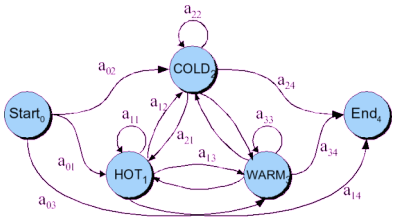
$A = a_{01} a_{02} \ldots a_{n1} \ldots a_{nn}$ — a **transition probability matrix** A, each a_{ij} representing the probability of moving from state i to state j, s.t. $\sum_{j=1}^{n} a_{ij} = 1 \quad \forall i$

q_0, q_{end} — a special **start and end state** which are not associated with observations.

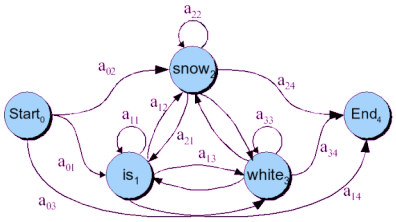
Markov Assumption: $P(q_i|q_1 \ldots q_{i-1}) = P(q_i|q_{i-1})$

HMMs more formally

- Markov chains
- A kind of weighted finite-state automaton



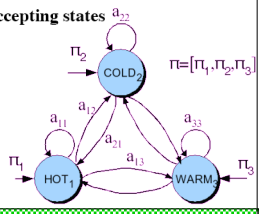
Another Markov chain



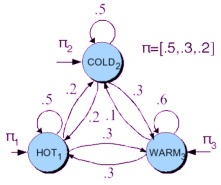
Another view of Markov chains

$\pi = \pi_1, \pi_2, \ldots, \pi_N$ — an **initial probability distribution** over states. π_i is the probability that the Markov chain will start in state i. Some states j may have $\pi_j = 0$, meaning that they cannot be initial states. Also, $\sum_{i=1}^{n} \pi_i = 1$

$QA = \{q_x, q_y \ldots\}$ — a set $QA \subset Q$ of legal **accepting states**



An example with numbers:



- What is probability of:
  - Hot hot hot hot
  - Cold hot cold hot

Hidden Markov Models

$Q = q_1 q_2 \ldots q_N$ — a set of N **states**

$A = a_{11} a_{12} \ldots a_{n1} \ldots a_{nn}$ — a **transition probability matrix** A, each a_{ij} representing the probability of moving from state i to state j, s.t. $\sum_{j=1}^{n} a_{ij} = 1 \quad \forall i$

$O = o_1 o_2 \ldots o_T$ — a sequence of T **observations**, each one drawn from a vocabulary $V = v_1, v_2, \ldots, v_V$.

$B = b_i(o_t)$ — A sequence of **observation likelihoods:**, also called **emission probabilities**, each expressing the probability of an observation o_t being generated from a state i.

q_0, q_F — a special **start state** and **end (final) state** which are not associated with observations, together with transition probabilities $a_{01} a_{02} .. a_{0n}$ out of the start state and $a_{1F} a_{2F} \ldots a_{nF}$ into the end state.

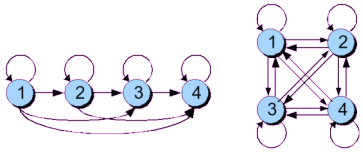
Hidden Markov Models

Markov Assumption: $P(q_i|q_1 \ldots q_{i-1}) = P(q_i|q_{i-1})$

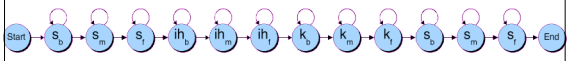
Output Independence Assumption: $P(o_i|q_1 \ldots q_i, \ldots, q_n, o_1, \ldots, o_i, \ldots, o_n) = P(o_i|q_i)$

Hidden Markov Models

- Bakis network
- Ergodic (fully-connected) network

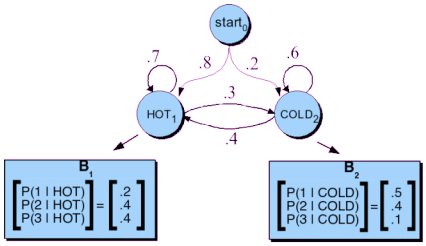


- Left-to-right network



The Jason Eisner task

- You are a climatologist in 2799 studying the history of global warming
- YOU can't find records of the weather in Baltimore for summer 2006
- But you do find Jason Eisner's diary
- Which records how many ice creams he ate each day.
- Can we use this to figure out the weather?
  - Given a sequence of observations O,
    - each observation an integer = number of ice creams eaten
    - Figure out correct hidden sequence Q of weather states (H or C) which caused Jason to eat the ice cream



HMMs more formally

- Three fundamental problems
  - Jack Ferguson at IDA in the 1960s

1) Given a specific HMM, determine likelihood of observation sequence.

2) Given an observation sequence and an HMM, discover the best (most probable) hidden state sequence

3) Given only an observation sequence, learn the HMM parameters (A, B matrix)

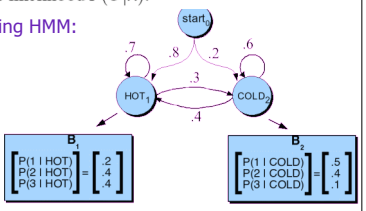
The Three Basic Problems for HMMs

- Problem 1 (**Evaluation**): Given the observation sequence $O=(o_1 o_2 \ldots o_T)$, and an HMM model $\Phi = (A,B)$, how do we efficiently compute $P(O|\Phi)$, the probability of the observation sequence, given the model
- Problem 2 (**Decoding**): Given the observation sequence $O=(o_1 o_2 \ldots o_T)$, and an HMM model $\Phi = (A,B)$, how do we choose a corresponding state sequence $Q=(q_1 q_2 \ldots q_T)$ that is optimal in some sense (i.e., best explains the observations)
- Problem 3 (**Learning**): How do we adjust the model parameters $\Phi = (A,B)$ to maximize $P(O|\Phi)$?

Problem 1: computing the observation likelihood

Computing Likelihood: Given an HMM $\lambda = (A, B)$ and an observation sequence O, determine the likelihood $P(O|\lambda)$.

- Given the following HMM:



- How likely is the sequence 3 1 3?

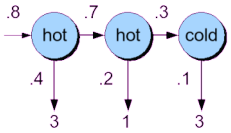
How to compute likelihood

- For a Markov chain, we just follow the states 3 1 3 and multiply the probabilities
- But for an HMM, we don't know what the states are!
- So let's start with a simpler situation.
- Computing the observation likelihood for a **given** hidden state sequence
  - Suppose we knew the weather and wanted to predict how much ice cream Jason would eat.
  - I.e. P(3 1 3 | H H C)

Computing likelihood for 1 given hidden state sequence

$$P(O|Q) = \prod_{i=1}^{n} P(o_i|q_i) \times \prod_{i=1}^{n} P(q_i|q_{i-1})$$

$$P(3\ 1\ 3|\text{hot hot cold}) = P(\text{hot}|\text{start}) \times P(\text{hot}|\text{hot}) \times P(\text{cold}|\text{hot}) \times P(3|\text{hot}) \times P(1|\text{hot}) \times P(3|\text{cold})$$



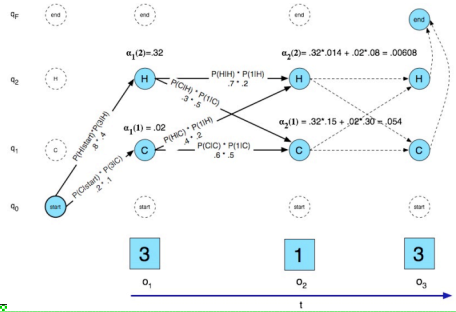
Computing total likelihood of 3 1 3

- We would need to sum over
  - Hot hot cold
  - Hot hot hot
  - Hot cold hot
  - ….
- How many possible hidden state sequences are there for this sequence?
- How about in general for an HMM with N hidden states and a sequence of T observations?
  - N^T
- So we can't just do separate computation for each hidden state sequence.

Instead: the Forward algorithm

- A kind of **dynamic programming** algorithm
  - Uses a table to store intermediate values
- Idea:
  - Compute the likelihood of the observation sequence
  - By summing over all possible hidden state sequences
  - But doing this efficiently
    - By folding all the sequences into a single **trellis**

The Forward Trellis



The forward algorithm

- Each cell of the forward algorithm trellis alpha$_t$(j)
  - Represents the probability of being in state j
  - After seeing the first t observations
  - Given the automaton
- Each cell thus expresses the following probabilty

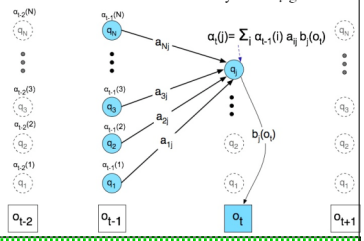
$$\alpha_t(j) = P(o_1, o_2 \ldots o_t, q_t = j|\lambda)$$

We update each cell

$\alpha_{t-1}(i)$ the **previous forward path probability** from the previous time step

a_{ij} the **transition probability** from previous state q_i to current state q_j

$b_j(o_t)$ the **state observation likelihood** of the observation symbol o_t given the current state j



The Forward Recursion

1. Initialization:

$$\alpha_1(j) = a_{0j} b_j(o_1) \quad 1 \le j \le N$$

2. Recursion (since states 0 and F are non-emitting):

$$\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} b_j(o_t); \quad 1 \le j \le N, 1 < t \le T$$

3. Termination:

$$P(O|\lambda) = \alpha_T(q_F) = \sum_{i=1}^{N} \alpha_T(i) a_{iF}$$

The Forward Algorithm

function FORWARD(*observations* of len T, *state-graph* of len N) **returns** *forward-prob*

create a probability matrix *forward*$[N+2,T]$
for each state s **from** 1 **to** N **do** ;initialization step
  $forward[s,1] \leftarrow a_{0,s} * b_s(o_1)$
for each time step t **from** 2 **to** T **do** ;recursion step
  for each state s **from** 1 **to** N **do**

$$forward[s,t] \leftarrow \sum_{s'=1}^{N} forward[s',t-1] * a_{s',s} * b_s(o_t)$$

$$forward[q_F,T] \leftarrow \sum_{i=1}^{N} forward[s,T] * a_{s,q_F} \quad \text{; termination step}$$

return $forward[q_F,T]$

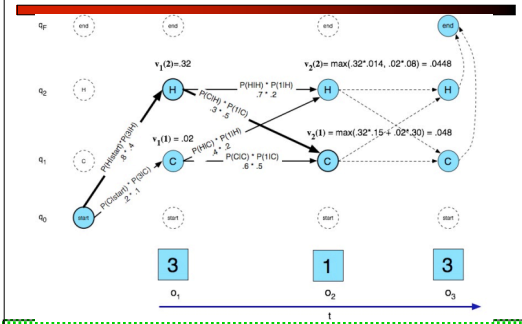
Decoding

- Given an observation sequence
  - 3 1 3
- And an HMM
- The task of the **decoder**
  - To find the best **hidden** state sequence
- Given the observation sequence O=(o_1o_2...o_T), and an HMM model Φ = (A,B), how do we choose a corresponding state sequence Q=(q_1q_2...q_T) that is optimal in some sense (i.e., best explains the observations)

Decoding

- One possibility:
  - For each hidden state sequence
    - HHH, HHC, HCH,
  - Run the forward algorithm to compute P(Φ |O)
- Why not?
  - N^T
- Instead:
  - The Viterbi algorithm
  - Is again a **dynamic programming** algorithm
  - Uses a similar trellis to the Forward algorithm

The Viterbi trellis



Viterbi intuition

- Process observation sequence left to right
- Filling out the trellis
- Each cell:

$$v_t(i) = P(q_0, q_1 \ldots q_{t-1}, o_1, o_2 \ldots o_t, q_t = i|\lambda)$$

$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i)\, a_{ij}\, b_j(o_t)$$

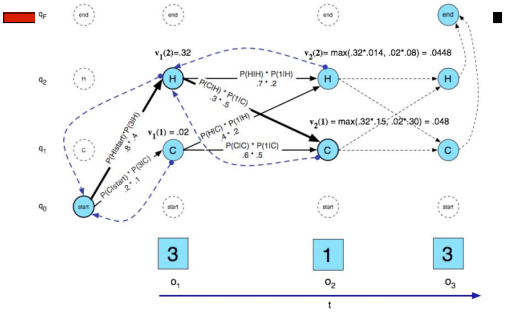
$v_{t-1}(i)$	the **previous Viterbi path probability** from the previous time step
a_{ij}	the **transition probability** from previous state q_i to current state q_j
$b_j(o_t)$	the **state observation likelihood** of the observation symbol o_t given the current state j

Viterbi Algorithm

function VITERBI(*observations* of len T,*state-graph* of len N) **returns** *best-path*

create a path probability matrix *viterbi*$[N+2,T]$
for each state s **from** 1 **to** N **do** ;initialization step
  viterbi$[s,1] \leftarrow a_{0,s} * b_s(o_1)$
  backpointer$[s,1] \leftarrow 0$
for each time step t **from** 2 **to** T **do** ;recursion step
  for each state s **from** 1 **to** N **do**
    viterbi$[s,t] \leftarrow \max_{s'=1}^{N}$ *viterbi*$[s',t-1] * a_{s',s} * b_s(o_t)$
    backpointer$[s,t] \leftarrow \text{argmax}_{s'=1}^{N}$ *viterbi*$[s',t-1] * a_{s',s}$
viterbi$[q_F,T] \leftarrow \max_{i=1}^{N}$ *viterbi*$[s,T] * a_{s,q_F}$; termination step
backpointer$[q_F,T] \leftarrow \text{argmax}_{i=1}^{N}$ *viterbi*$[s,T] * a_{s,q_F}$; termination step
return the backtrace path by following backpointers to states back in time from *backpointer*$[q_F,T]$

Viterbi backtrace



Viterbi Recursion

1. **Initialization:**

$$v_1(j) = a_{0j} b_j(o_1) \quad 1 \le j \le N$$
$$bt_1(j) = 0$$

2. **Recursion** (recall states 0 and q_F are non-emitting):

$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i)\, a_{ij}\, b_j(o_t); \quad 1 \le j \le N, 1 < t \le T$$
$$bt_t(j) = \text{argmax}_{i=1}^{N} v_{t-1}(i)\, a_{ij}\, b_j(o_t); \quad 1 \le j \le N, 1 < t \le T$$

3. **Termination:**

The best score: $P* = v_t(q_F) = \max_{i=1}^{N} v_T(i) * a_{i,F}$

The start of backtrace: $q_T* = bt_T(q_F) = \text{argmax}_{i=1}^{N} v_T(i) * a_{i,F}$

Why "Dynamic Programming"

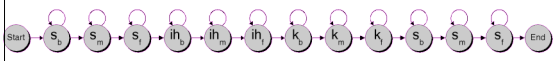
"I spent the Fall quarter (of 1950) at RAND. My first task was to find a name for multistage decision processes. An interesting question is, Where did the name, dynamic programming, come from? The 1950s were not good years for mathematical research. We had a very interesting gentleman in Washington named Wilson. He was Secretary of Defense, and he actually had a pathological fear and hatred of the word, research. I'm not using the term lightly; I'm using it precisely. His face would suffuse, he would turn red, and he would get violent if people used the term, research, in his presence. You can imagine how he felt, then, about the term, mathematical. The RAND Corporation was employed by the Air Force, and the Air Force had Wilson as its boss, essentially. Hence, I felt I had to do something to shield Wilson and the Air Force from the fact that I was really doing mathematics inside the RAND Corporation. What title, what name, could I choose? In the first place I was interested in planning, in decision making, in thinking. But planning, is not a good word for various reasons. I decided therefore to use the word, "programming". I wanted to get across the idea that this was dynamic, this was multistage, this was time-varying I thought, lets kill two birds with one stone. Lets take a word that has an absolutely precise meaning, namely dynamic, in the classical physical sense. It also has a very interesting property as an adjective, and that is its impossible to use the word, dynamic, in a pejorative sense. Try thinking of some combination that will possibly give it a pejorative meaning. Its impossible. Thus, I thought dynamic programming was a good name. It was something not even a Congressman could object to. So I used it as an umbrella for my activities." Richard Bellman, "Eye of the Hurrican: an autobiography" 1984.

Thanks to Chen, Picheny, Eide, Nock

HMMs for Speech

- We haven't yet shown how to learn the A and B matrices for HMMs; we'll do that later today or possibly on Monday
- But let's return to think about speech

Reminder: a word looks like this:

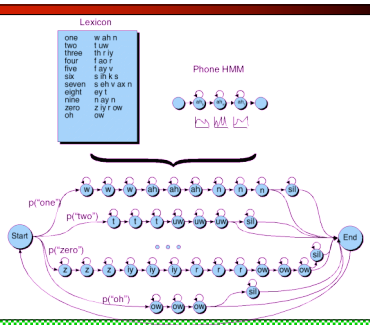


$Q = q_1 q_2 \ldots q_N$ — a set of **states** corresponding to **subphones**

$A = a_{01} a_{02} \ldots a_{n1} \ldots a_{nn}$ — a **transition probability matrix** A, each a_{ij} representing the probability for each subphone of taking a **self-loop** or going to the next subphone. Together, Q and A implement a **pronunciation lexicon**, an HMM state graph structure for each word that the system is capable of recognizing.

$B = b_i(o_t)$ — A set of **observation likelihoods:**, also called **emission probabilities**, each expressing the probability of a cepstral feature vector (observation o_t) being generated from subphone state i.

HMM for digit recognition task



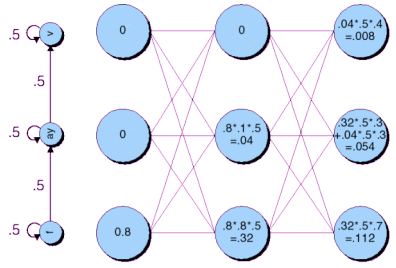
The Evaluation (forward) problem for speech

- The observation sequence O is a series of MFCC vectors
- The hidden states W are the phones and words
- For a given phone/word string W, our job is to evaluate P(O|W)
- Intuition: how likely is the input to have been generated by just that word string W

Evaluation for speech: Summing over all different paths!

- f ay ay ay v v v v
- f f ay ay ay v v v
- f f f f ay ay ay v
- f f ay ay ay ay ay v
- f f ay ay ay ay ay ay v
- f f ay v v v v v v v

The forward lattice for "five"

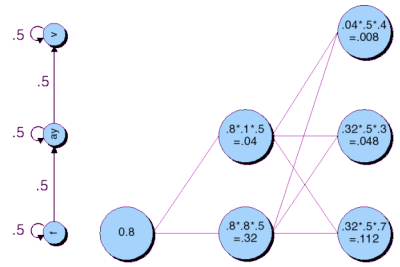


The forward trellis for "five"

V	0	0	0.008	0.0093	0.0114	0.00703	0.00345	0.00306	0.00206	0.00117
AY	0	0.04	0.054	0.0664	0.0355	0.016	0.00676	0.00208	0.000532	0.000109
F	0.8	0.32	0.112	0.0224	0.00448	0.000896	0.000179	4.48e-05	1.12e-05	2.8e-06

Time	1	2	3	4	5	6	7	8	9	10
B	*f 0.8* *ay 0.1* *v 0.6* *p 0.4* *iy 0.1*	*f 0.8* *ay 0.1* *v 0.6* *p 0.4* *iy 0.1*	*f 0.7* *ay 0.3* *v 0.4* *p 0.2* *iy 0.3*	*f 0.4* *ay 0.8* *v 0.3* *p 0.1* *iy 0.6*	*f 0.4* *ay 0.8* *v 0.3* *p 0.1* *iy 0.6*	*f 0.4* *ay 0.8* *v 0.3* *p 0.1* *iy 0.6*	*f 0.4* *ay 0.8* *v 0.3* *p 0.1* *iy 0.6*	*f 0.5* *ay 0.6* *v 0.6* *p 0.1* *iy 0.5*	*f 0.5* *ay 0.5* *v 0.8* *p 0.3* *iy 0.5*	*f 0.5* *ay 0.4* *v 0.9* *p 0.3* *iy 0.4*

Viterbi trellis for "five"

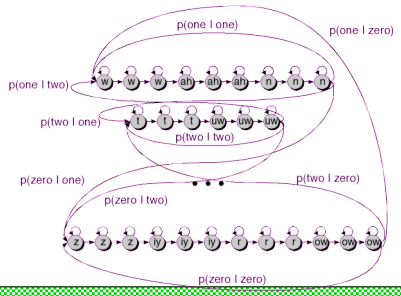


Viterbi trellis for "five"

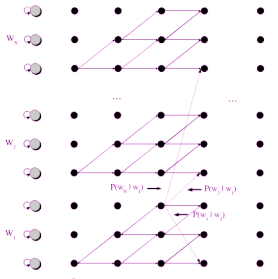
V	0	0	0.008	0.0072	0.00672	0.00403	0.00188	0.00161	0.000667	0.000493
AY	0	0.04	0.048	0.0448	0.0269	0.0125	0.00538	0.00167	0.000428	8.78e-05
F	0.8	0.32	0.112	0.0224	0.00448	0.000896	0.000179	4.48e-05	1.12e-05	2.8e-06

Time	1	2	3	4	5	6	7	8	9	10
B	*f 0.8* *ay 0.1* *v 0.6* *p 0.4* *iy 0.1*	*f 0.8* *ay 0.1* *v 0.6* *p 0.4* *iy 0.1*	*f 0.7* *ay 0.3* *v 0.4* *p 0.2* *iy 0.3*	*f 0.4* *ay 0.8* *v 0.3* *p 0.1* *iy 0.6*	*f 0.4* *ay 0.8* *v 0.3* *p 0.1* *iy 0.6*	*f 0.4* *ay 0.8* *v 0.3* *p 0.1* *iy 0.6*	*f 0.4* *ay 0.8* *v 0.3* *p 0.1* *iy 0.6*	*f 0.5* *ay 0.6* *v 0.6* *p 0.1* *iy 0.5*	*f 0.5* *ay 0.5* *v 0.8* *p 0.3* *iy 0.5*	*f 0.5* *ay 0.4* *v 0.9* *p 0.3* *iy 0.4*

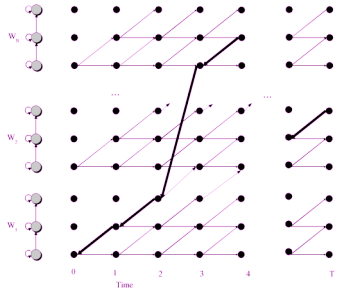
Search space with bigrams

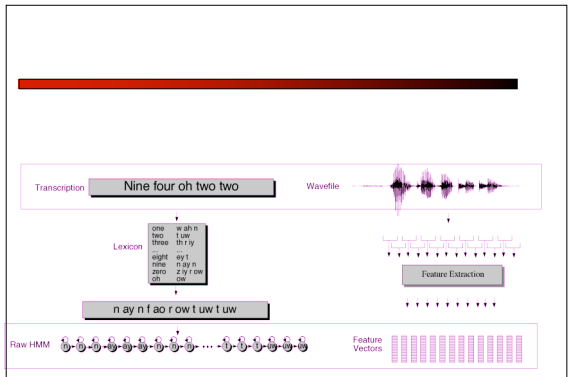


Viterbi trellis with 2 words and uniform LM



Viterbi backtrace





Evaluation

- How to evaluate the word string output by a speech recognizer?

Word Error Rate

- Word Error Rate =

100 (Insertions+Substitutions + Deletions)

Total Word in Correct Transcript

Aligment example:

REF: portable **** PHONE UPSTAIRS last night so

HYP: portable FORM OF STORES last night so

Eval I S S

WER = 100 (1+2+0)/6 = 50%

NIST sctk-1.3 scoring softare: Computing WER with sclite

- http://www.nist.gov/speech/tools/
- Sclite aligns a hypothesized text (HYP) (from the recognizer) with a correct or reference text (REF) (human transcribed)

```
id: (2347-b-013)
Scores: (#C #S #D #I) 9 3 1 2
REF:  was an engineer SO I   i was always with **** **** MEN UM   and they
HYP:  was an engineer ** AND i was always with THEM THEY ALL THAT and they
Eval:                 D  S                    I    I    S   S
```


Sclite output for error analysis

```
CONFUSION PAIRS                 Total                (972)
                                With >=  1 occurances (972)

   1:   6  ->  (%hesitation) ==> on
   2:   6  ->  the ==> that
   3:   5  ->  but ==> that
   4:   4  ->  a ==> the
   5:   4  ->  four ==> for
   6:   4  ->  in ==> and
   7:   4  ->  there ==> that
   8:   3  ->  (%hesitation) ==> and
   9:   3  ->  (%hesitation) ==> the
  10:   3  ->  (a-) ==> i
  11:   3  ->  and ==> i
  12:   3  ->  and ==> in
  13:   3  ->  are ==> there
  14:   3  ->  as ==> is
  15:   3  ->  have ==> that
  16:   3  ->  is ==> this
```


Sclite output for error analysis

```
  17:   3  ->  it ==> that
  18:   3  ->  mouse ==> most
  19:   3  ->  was ==> is
  20:   3  ->  was ==> this
  21:   3  ->  you ==> we
  22:   2  ->  (%hesitation) ==> it
  23:   2  ->  (%hesitation) ==> that
  24:   2  ->  (%hesitation) ==> to
  25:   2  ->  (%hesitation) ==> yeah
  26:   2  ->  a ==> all
  27:   2  ->  a ==> know
  28:   2  ->  a ==> you
  29:   2  ->  along ==> well
  30:   2  ->  and ==> it
  31:   2  ->  and ==> we
  32:   2  ->  and ==> you
  33:   2  ->  are ==> i
  34:   2  ->  are ==> were
```

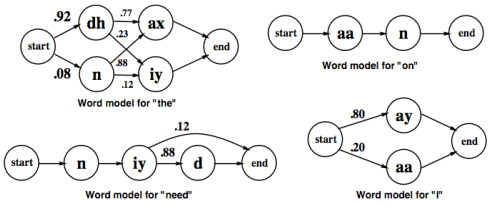
Better metrics than WER?

- WER has been useful
- But should we be more concerned with meaning ("semantic error rate")?
  - Good idea, but hard to agree on
  - Has been applied in dialogue systems, where desired semantic output is more clear

Summary: ASR Architecture

- Five easy pieces: ASR Noisy Channel architecture
  1) Feature Extraction:
     39 "MFCC" features
  2) Acoustic Model:
     Gaussians for computing p(o|q)
  3) Lexicon/Pronunciation Model
     - HMM: what phones can follow each other
  4) Language Model
     - N-grams for computing $p(w_i|w_{i-1})$
  5) Decoder
     - Viterbi algorithm: dynamic programming for combining all these to get word sequence from speech!

ASR Lexicon: Markov Models for pronunciation



Summary

- Speech Recognition Architectural Overview
- Hidden Markov Models in general
  - Forward
  - Viterbi Decoding
- Hidden Markov models for Speech
- Evaluation