Speech recognition and synthesis

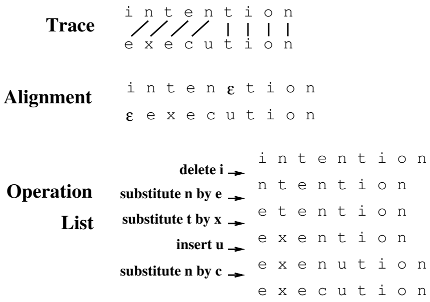
Introduction

Two technologies are needed to make the HMM framework practical

- Decoder technology to find the $\underset{Words}{argmax}\ P(Observation|Words) \cdot P(Words)$
- Determining the stochastic parameters of the HMM state automaton, i.e. training

Many pictures (and their copyrights) are from [Jurafsky and Martin(2000)]

Dynamic programming



Look for best alignment: Minimum edit distance

- Delete
- Insert
- Substitute

Dynamic programming

function MIN-EDIT-DISTANCE(*target*, *source*) **returns** *min-distance*

```
n ← LENGTH(target)
m ← LENGTH(source)
Create a distance matrix distance[n+1,m+1]
distance[0,0] ← 0
for each column i from 0 to n do
   for each row j from 0 to m do
      distance[i,j] ← MIN( distance[i−1,j] + ins-cost(target_i),
                           distance[i−1,j−1] + subst-cost(source_j, target_i),
                           distance[i,j−1] + del-cost(source_j))
```

Fill a matrix with cumulative edit distances, $\mathit{distance}[i,j] = \min$ of

- $\mathit{distance}[i-1,j] + \text{insert-cost}(\mathit{target}_i)$
- $\mathit{distance}[i-1,j-1] + \text{substitution-cost}(\mathit{source}_j, \mathit{target}_i)$
- $\mathit{distance}[i,j-1] + \text{deletion-cost}(\mathit{source}_j)$

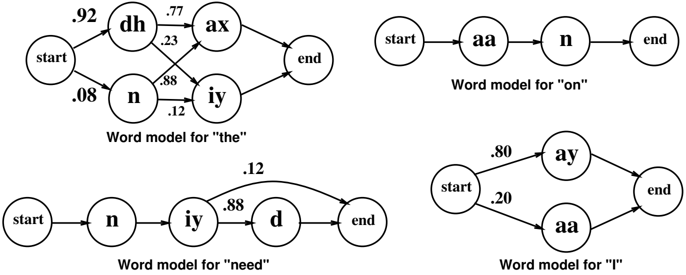
Dynamic programming

n	9	10	11	10	11	12	11	10	9	**8**
o	8	9	10	9	10	11	10	9	**8**	9
i	7	8	9	8	9	10	9	**8**	9	10
t	6	7	8	7	8	9	**8**	9	10	11
n	5	6	7	6	7	**8**	9	10	11	12
e	4	5	6	**5**	**6**	7	8	9	10	11
t	3	4	**5**	6	7	8	9	10	11	12
n	2	3	**4**	5	6	7	8	8	10	11
i	1	**2**	3	4	5	6	7	8	9	10
#	**0**	1	2	3	4	5	6	7	8	9
	#	e	x	e	c	u	t	i	o	n

Trace back the choices of the minimal distance (bold numbers)

- This finds the globally minimal cost path
- Full search unwieldy for large and complex matrices
- In general, searches are pruned to exclude paths that deviate far from the diagonal: Beam search

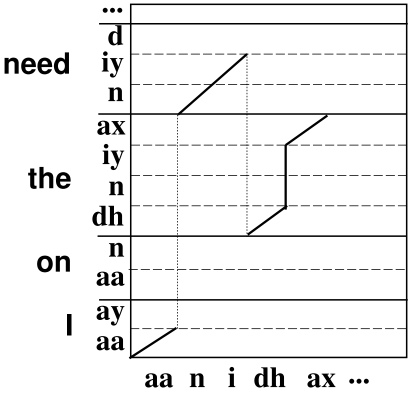
Viterbi algorithm



Simplified pronunciation networks [Jurafsky and Martin(2000)]

- Each word is modeled as a Finite State Machine
- Individual phoneme HMMs are trained from a corpus that does not contain all the words
- A pronunciation dictionary contains all word models
- Transition probabilities are "trained" from a transcribed speech corpus

Viterbi algorithm



- Whole sequence on **X** axis
- All word models on the other axis
- Switch to (any) new word after reaching the end of the current word
- Word switching cost based on the language model

Viterbi algorithm result *"for I need a"* [Jurafsky and Martin(2000)]

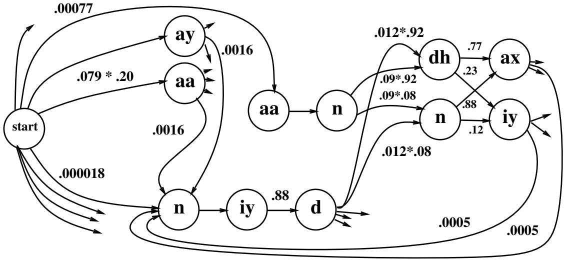
Viterbi algorithm

I need	0.0016	need need	0.000047	# Need	0.000018
I the	0.00018	need the	0.012	# The	0.016
I on	0.000047	need on	0.000047	# On	0.00077
I I	0.039	need I	0.000016	# I	0.079
the need	0.00051	on need	0.000055		
the the	0.0099	on the	0.094		
the on	0.00022	on on	0.0031		
the I	0.00051	on I	0.00085		

Bigram probabilities [Jurafsky and Martin(2000)]

- Word switching in Viterbi searches uses probabilities
- Switch to a new word with bigram probability cost
- Does not work with trigram probabilities

Viterbi algorithm



Single pronunciation automaton for *I*, *need*, *on*, and *the*

[Jurafsky and Martin(2000)]

- Bigram probabilities connect the word models
- Merge **start** and **end** states of connected words
- Need for *pruning* is apparent

Viterbi algorithm

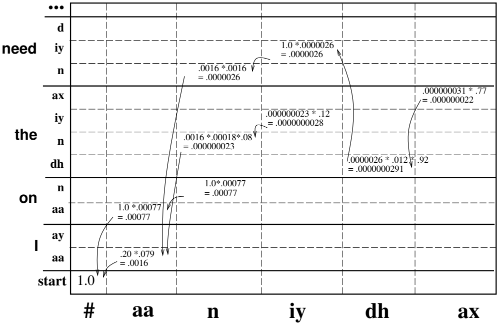
```
function VITERBI(observations of len T,state-graph) returns best-path

  num-states ← NUM-OF-STATES(state-graph)
  Create a path probability matrix viterbi[num-states+2,T+2]
  viterbi[0,0] ← 1.0
  for each time step t from 0 to T do
    for each state s from 0 to num-states do
      for each transition s' from s specified by state-graph
        new-score ← viterbi[s, t] * a[s,s'] * b_s'(o_t)
        if ((viterbi[s',t+1] = 0) || (new-score > viterbi[s', t+1]))
          then
                viterbi[s', t+1] ← new-score
                back-pointer[s', t+1] ← s
  Backtrace from highest probability state in the final column of viterbi[] and
return path
```

Extended version of the edit distance [Jurafsky and Martin(2000)]

- $a[s, s'] = P(s \rightarrow s')$
- $b_{s'}(o_t) = P(o_t | s')$

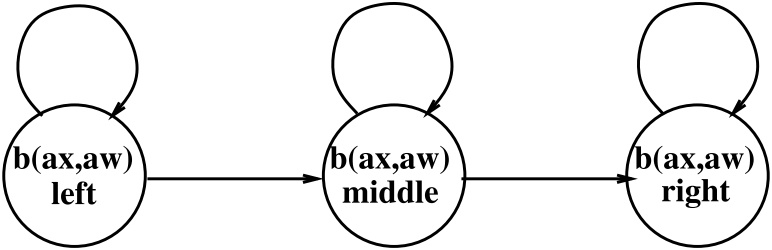
Viterbi algorithm



Individual state columns in Viterbi algorithm [Jurafsky and Martin(2000)]

- The actual entries for the Automaton
- Note the problems for a 20,000 word dictionary

Viterbi algorithm: Subphones revisited [Jurafsky and Martin(2000)]



Use structured, context sensitive phone units

- Single phone units perform bad due to coarticulation
- *Begin* differs from *End* (eg, /d/)
- 60 context dependent triphones $\Rightarrow 60^3 = 216000$ models
- Cluster contexts,eg, on manner and place of articulation

Other approaches to decoding: Introduction

The standard HMM model has limitations

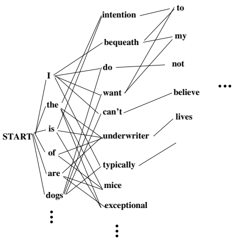
- Viterbi decoder penalizes multiple pronunciations
- Viterbi decoder does not work for anything more complex than bigram
- It is not possible to include other linguistic knowledge
  - Phoneme duration (HMM have a Poison distribution)
  - Intonation
  - Semantics
  - Speaker identification
  - Expressive speech tags
  - Task related knowledge

Other approaches to decoding

Two stage N-best decoding [Jurafsky and Martin(2000)]

- Keep N-best utterance list or word lattice
- Rescore the probabilities with the extra knowledge
  - A trigram or higher grammar
  - Phoneme duration probability
  - Parallel Intonation and Accent detector (HMM)
  - Include semantic or task related knowledge
  - Multiple speakers and expressive speech tags
- Look up best path through rescored word lattice

Other approaches to decoding: A^*



Stack, or A^*, decoding [Jurafsky and Martin(2000)]

- Viterbi uses best path upto position t to get to $t + 1$
- A^* uses complete forward algorithm (exact likelihoods)
- A^* searches potential utterances best-first

Other approaches to decoding: A^*

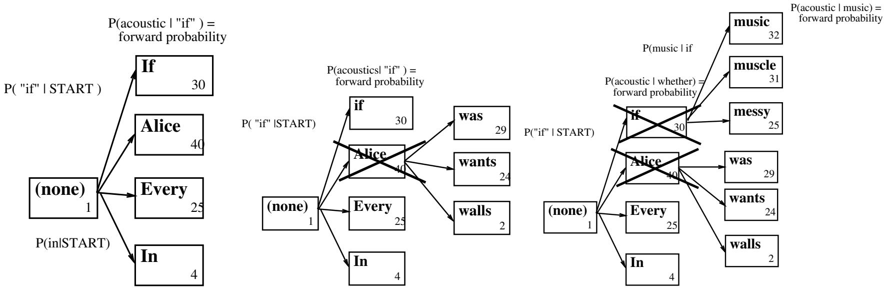
function STACK-DECODING() **returns** *min-distance*

Initialize the priority queue with a null sentence.
Pop the best (highest score) sentence s off the queue.
If (s is marked end-of-sentence (EOS)) output s and terminate.
Get list of candidate next words by doing fast matches.
For each candidate next word w:
  Create a new candidate sentence $s+w$.
  Use forward algorithm to compute acoustic likelihood L of $s+w$
  Compute language model probability P of extended sentence $s+w$
  Compute "score" for $s+w$ (a function of L, P, and ???)
  if (end-of-sentence) set EOS flag for $s+w$.
  Insert $s+w$ into the queue together with its score and EOS flag

Stack decoding [Jurafsky and Martin(2000)]

- At each point, the A^* looks for the most likely next word
- Acoustic likelihood is part of the criterium
- Use the forward probability of preceding words

Other approaches to decoding: A^*



If music be the food of love [Jurafsky and Martin(2000)]

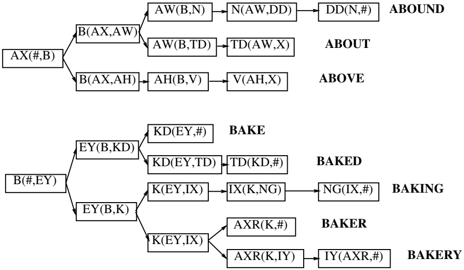
- *"Start Alice"* has highest score: 40
- *"Start if"* has highest score: 30
- *"Start if music"* has highest score: 32

Other approaches to decoding: A^*

Remarks

- Use fast match heuristics for selecting next words
- Longer utterances have lower probabilities, score should correct for this
- A^* evaluation function: $f^*(p) = g(p) + h^*(p)$
- $g(partial\ path) = P(O|Words) \cdot P(Words)$, i.e. the likelihood until now
- $h^*(p)$ something that correlates with number of words in the rest of the utterance
- Defining a good $h^*(p)$ is an interesting (unsolved) problem

Other approaches to decoding: A^* fast match



A tree structured lexicon from SPHINX [Gouvêa()][Jurafsky and Martin(2000)]

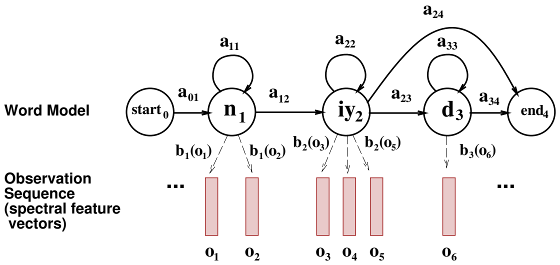
- Need to get forward probabilities of potential continuations *fast*
- Tree lexicon shares forward probabilities between words
- Allows early pruning of search trees

Training acoustic models: Introduction

Determine $P(Observation|Words)$, i.e. the transition probability between phone states a_{ij} and the acoustic likelihood of the speech vectors $b_j(o_k)$

- Large, "transcribed" speech corpus (on text level)
- Coverage of speakers and language types
- Recorded under the same conditions as intended use, eg, over the phone or in a driving car
- Use the same microphone etc.
- Using a simulated task (Wizard of Oz or Green curtain) to elicit the same kind of speech

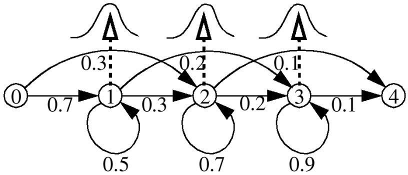
Training acoustic models



If all states were known [Jurafsky and Martin(2000)]

- $a_{ij} = \dfrac{\#S_{ij}}{\#S_{i*}}$ (count transitions and states)
- $b_i(O_k) = \dfrac{\#(O_k \& S_i)}{\#S_i}$ (for discrete O_k)

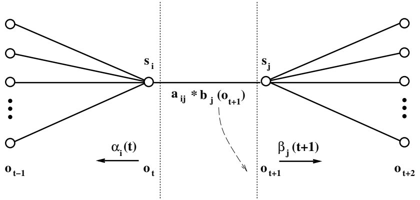
Training acoustic models



If observations are continuous vectors [SPH()]

- $b_i(O_t) \Rightarrow N\{\hat{\mu}_i, \hat{\Sigma}_i\}$
- $\hat{\mu}_i = \frac{1}{T_i} \sum_{t=1}^{T_i} O_t$
- $\hat{\Sigma}_i = \frac{1}{T_i} \sum_{t=1}^{T_i} [(O_t - \hat{\mu}_i)'(O_t - \hat{\mu}_i)]$

Training acoustic models



States have to be estimated. Use an iterative procedure App D [Jurafsky and Martin(2000)]

- Run the recognizer on the corpus with the known words
- Calculate $\hat{a}_{ij} = \frac{expected\ \#S_i \rightarrow S_j}{expected\ \#S_i \rightarrow S_*}$
- Calculate $\hat{b}_j(v_k) = \frac{expected\ \#S_j\ observing\ v_k}{expected\ \#S_j}$
- Update all values and start again

FLOSS resources

Free and Open Source ASR systems

- SPHINX (CMU) [Gouvêa()] [Singh(2005)]
- CMU Statistical Language Modeling Toolkit [Rosenfeld()]
- CMU Pronouncing Dictionary [Lenzo()]
- Internet-Accessible Speech Recognition Technology project (ISIP, Mississippi State University) [ISIP(2004)]
- Open Mind Speech [Valin()]

Assignment: Week 8 Statistical Language Models

Construct your own language model

- Download texts from the internet, eg, [Project Gutenberg(2005)]
- Use a single author or a single genre
- Use `--help` to see instructions of the programs
- Construct unigram and bigram word tables with *Ngramcount.pl*
  `http://www.fon.hum.uva.nl/rob/Courses/Taaltechnologien/Ngramcount.pl`
- perl Ngramcount.pl 1 <filename1> <filename2> ... > unigramtable.txt
- perl Ngramcount.pl 2 <filename1> <filename2> ... > bigramtable.txt
- Inspect the table files. What are the most frequent words and bigrams?
- Calculate the probabilities of sentences with *ngramprobability.pl*
  `http://www.fon.hum.uva.nl/rob/Courses/Taaltechnologien/ngramprobability.pl`
- perl ngramprobability.pl –count 5 –verbose bigramtable.txt "<sentence>"
- Enter some sentences and inspect the resulting probabilities
- Experiment with the `--count` option. Try `--count -1` on a sentence that contains unknown word combinations

Further Reading I

P. Boersma.
Praat, a system for doing phonetics by computer.
Glot International, 5:341–345, 2001.
URL http://www.Praat.org/.

P. Boersma and D. Weenink.
Praat 4.2: doing phonetics by computer.
Computer program: http://www.Praat.org/, 2004.
URL http://www.Praat.org/.

CSLU.
CSLU Toolkit.
Web.
URL http://cslu.cse.ogi.edu/toolkit/index.html.

FSF.
GNU General Public License.
Web, June 1991.
URL http://www.gnu.org/licenses/gpl.html.

Joshua T. Goodman.
A bit of progress in language modeling.
Computer Speech and Language, 15:403–434, 2001.
URL http://arxiv.org/abs/cs.CL/0108005.
URL is extended preprint.

Further Reading II

E. Gouvêa.
The CMU Sphinx Group Open Source Speech Recognition Engines.
Web.
URL http://cmusphinx.sourceforge.net/html/cmusphinx.php.

ISIP.
The Mississippi State ISIP public domain speech recognizer.
Web, August 2004.
URL http://www.cavs.msstate.edu/hse/ies/projects/speech/index.html.

Daniel Jurafsky and James H. Martin.
Speech and Language Processing.
Prentice-Hall, 2000.
ISBN 0-13-095069-6.
URL http://www.cs.colorado.edu/~martin/slp.html.
Updates at http://www.cs.colorado.edu/

Kevin Lenzo.
The CMU Pronouncing Dictionary.
Web.
URL http://www.speech.cs.cmu.edu/SLM_info.html.

Project Gutenberg.
Project gutenberg free ebook library.
Web, 2005.
URL http://www.gutenberg.org/.

Further Reading III

Roni Rosenfeld.
The CMU Statistical Language Modeling (SLM) Toolkit.
Web.
URL http://www.speech.cs.cmu.edu/SLM_info.html.

Rita Singh.
Robust group's open source tutorial learning to use the cmu sphinx automatic speech recognition system.
Web, 2005.
URL http://www.cs.cmu.edu/~robust/Tutorial/opensource.html.

Manual for the Sphinx-III recognition system.
SPHINX-CMU.
URL http://fife.speech.cs.cmu.edu/sphinxman/.

Paul A. Taylor, S. King, S. D. Isard, and H. Wright.
Intonation and dialogue context as constraints for speech recognition.
Language and Speech, 41:493–512, 1998.
URL http://www.cstr.ed.ac.uk/downloads/publications/1998/Taylor_1998_b.pdf.

Jean-Marc Valin.
Open mind speech.
Web.
URL http://freespeech.sourceforge.net/.

Xue Wang.
incorporating knowledge on segmental duration in hmm-based continuous speech recognition.
PhD thesis, LOT Netherlands Graduate School of Linguistics, 04 1997.
URL http://www.fon.hum.uva.nl/wang/ThesisWangXue/TOCINDEX.html.

Appendix A

Copyright License