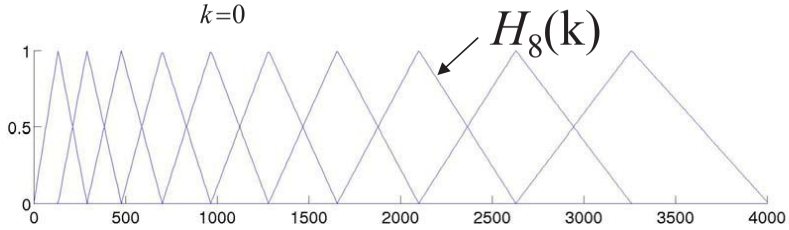
Speech Recognition

- Humans are capable of
  - Speaker independent recognition
  - Continuous, spontaneous speech recognition
  - Recognition in noisy environments.
  - Recognizing words from a large vocabulary.
- Machines are capable of
  - Speaker independent, continuous, *small vocabulary* recognition.
  - Large vocabulary, *speaker dependent*, *isolated word* recognition.
- Machines have problems with robustness in several aspects
  - Variations in speech style (conversational speech - reading a book).
  - Noisy environments (speech enhancement helps).
  - Acoustic differences between speakers.

Mel-frequency cepstrum coefficients (MFCCs)

- Typically, vectors of MFCCs are extracted from 25 ms speech frames every 10 ms.
  - First the DFT of the frame is calculated, $X(k)$
  - Next, the squared magnitudes are filtered by a mel-filter bank

$$S(m) = \log \sum_{k=0}^{N-1} |X(k)|^2 \, H_m(k), \quad 1 \le m \le M$$



  - The MFCC is the DCT of the log-energies $S(m)$

$$c(n) = \sum_{m=1}^{M} S(m) \cos(\pi n (m - 1/2)/M), \quad 0 \le n \le M - 1$$

- M is 24-40, but only the first 13 coefficients are used
- Popular choice in both speaker and speech recognition
  - LP cepstral coefficients not so common nowadays

A Statistical Framework for Speech Recognition

- The problem of finding the best word sequence W given an observed sequence of features, X (LPC cepstrum vectors, MFCCs), can be formulated as a MAP estimate

$$W_* = \arg\max_W P(W \mid X)$$

Bayes rule yields

$$W_* = \arg\max_W \frac{P(X \mid W)P(W)}{P(X)} = \arg\max_W P(X \mid W)P(W)$$

- The distribution *P(X|W)* describes how a sequence of words generates a sequence of feature vectors. Modeling *P(X|W)* is called *acoustic modeling*.
- The distribution *P(W)* provides a grammar for a language via probabilities of sequences of words. Modeling *P(W)* is called *language modeling*.

Language Modeling

- The probability P(W) can be expressed

$$P(W) = P(W_1)\prod_{i=2}^{T} P(W_i \mid W_{i-1}, W_{i-2}, \ldots, W_1)$$

- *N*-grams approximate the above by

$$P(W) = P(W_1)\prod_{i=2}^{T} P(W_i \mid W_{i-1}, \ldots, W_{i-N-1})$$

Unigrams: $P(W) = P(W_1)\prod_{i=2}^{T} P(W_i)$

Bigrams: $P(W) = P(W_1)\prod_{i=2}^{T} P(W_i \mid W_{i-1})$

Trigrams: $P(W) = P(W_1)\prod_{i=2}^{T} P(W_i \mid W_{i-1}, W_{i-2})$

- *N*-gram models are estimated from large text databases (newspapers, books).

Acoustic Modeling

- Simplifications of P(X|W) are needed:

$$P(X \mid W) \approx \prod_{i=1}^{K} P(X^{(i)} \mid W_i)$$

One word

Assumes no coarticulation between words. Consider "got you" (gottcha), and "are you"…

- Models of $P(X^{(i)} \mid W_i)$ are called "whole-word" models.
- Further simplifications to phoneme-based word models

$$P(X^{(i)} \mid W_i) \approx \prod_{j=1}^{M} P(X^{(i,j)} \mid p_{j,i})$$

Assumes no coarticulation between phonemes.

where the acoustic realization of a word is constructed from a sequence of independent phones. A dictionary provides the phone sequence

$$W_i = (p_{1,i}, p_{2,i}, \ldots, p_{M,i})$$

- HMMs are often used to model $P(X^{(i)} \mid W_i)$ and $P(X^{(i,j)} \mid p_{j,i})$

Modeling Coarticulations

- For whole word models:

$$P(X \mid W) \approx \prod_{i=1}^{K} P(X^{(i)} \mid W_i, W_{i-1}, W_{i+1})$$

- For phoneme models:

$$P(X^{(i)} \mid W_i) \approx \prod_{j=1}^{M} P(X^{(i,j)} \mid p_{j,i}, p_{j-1,i}) \qquad \text{Diphone model}$$

$$P(X^{(i)} \mid W_i) \approx \prod_{j=1}^{M} P(X^{(i,j)} \mid p_{j,i}, p_{j-1,i}, p_{j+1,i}) \qquad \text{Triphone model}$$

- Still the dictionary provides the (di,tri)phone sequence

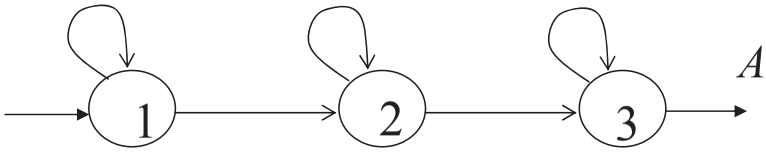
  Example: DRAMA /D R AA M AH/

  Triphone sequence: /sil **D** R/, /D **R** AA/, /R **AA** M/, /AA **M** AH/, /M **AH** sil/

- Drawback: Increased number of models

Left-to-Right HMMs

- A left-to right model topology is used for both whole-word, and phoneme models.



$$A = \begin{pmatrix} 0.6 & 0.4 & 0 & 0 \\ 0 & 0.3 & 0.7 & 0 \\ 0 & 0 & 0.5 & 0.5 \\ 0.2 & 0 & 0 & 0.8 \end{pmatrix}$$

- The number of states in a whole-word model is based on
  - the length of the word (15-25 states per second), or
  - the number of phonemes in the words.
- 3-5 states per phoneme.
- Training requires around 100 acoustic realizations per word/phoneme.

Why HMMs in Speech Recognition?

- Markov models allow for variations in speaking rate (how fast the speaker talks)
  - Inherent in the model since we move to the next state with a probability, not for certain
- Hidden Markov models allow for a statistical variation in the acoustic realization of word sequences
  - The output distributions model this variation
- We need not know the exact boundaries between words/phonemes in the observed feature sequence
  - Even though it may seem like that from our statistical framework
- Combining the language and acoustic model yields a (super) HMM
- Efficient search algorithms exist for finding the best word sequence

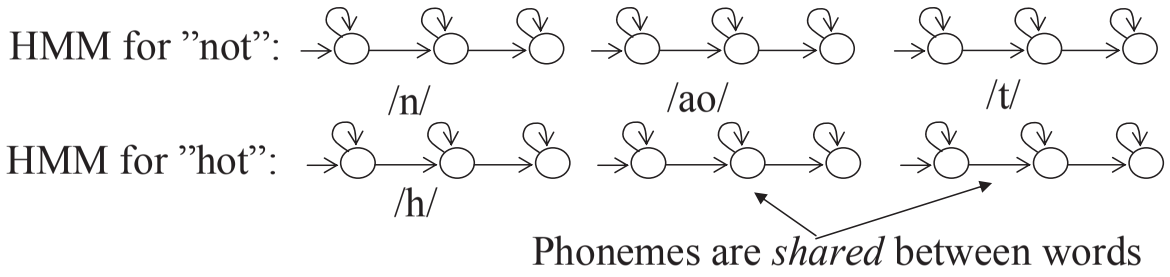
HMMs Trained on Whole-Words

- Whole-word HMMs are used in both isolated and continuous speech recognition applications.
- Whole word HMMs model coarticulation effects accurately
  - Within words, each phoneme is trained in the proper context
  - Across word boundaries.
    ”got you”, ”are you” –> two different models of ”you”
- If the vocabulary is large (>1000), the training complexity becomes prohibitively high.
- For small vocabulary tasks (like digit recognition, Swebus booking), whole word HMMs are accurate and trainable.
- For large vocabulary tasks, we say that whole-word HMMs are not generalizable (trainable) => inaccurate.

Phoneme HMMs, I

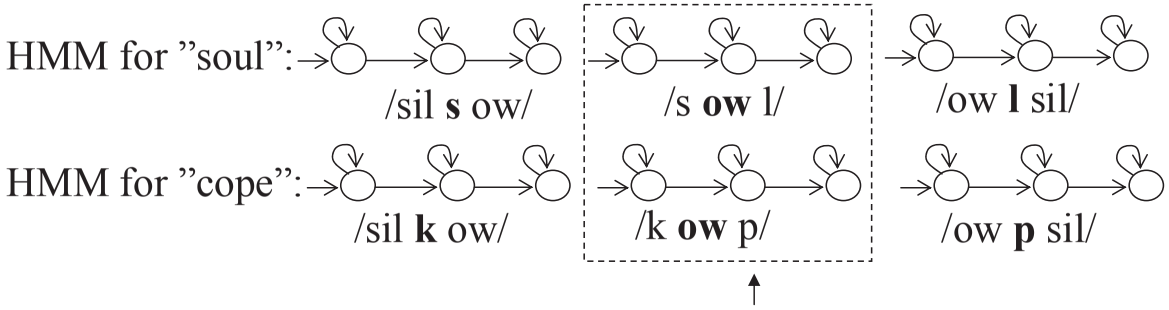
- Phoneme HMMs are used in both isolated and continuous speech recognition applications.
- 44 HMMs needed independently of vocabulary size (around 44 phonemes in English).
- Word models are constructed from phoneme HMMs by concatenating according to a dictionary

acid	/ae/ /s/ /ih/ /d/	hot	/h/ /ao/ /t/
not	/n/ /ao/ /t/	no	/n/ /ow/
use	/y/ uw/ /z/	soul	/s/ /ow/ /l/
cope	/k/ /ow/ /p/		



Phoneme HMMs, II

- Phoneme HMMs are generalizable (trainable), both towards larger vocabulary, and towards different speakers.
- They are less accurate than whole word models since they assume that each phoneme sounds the same in every context.
- *Triphone* HMMs take into account what comes before and after the phoneme: soul /s/ /ow/ /l/, cope /k/ /ow/ /p/



Two different realizations of the phoneme /ow/, i.e., two allophones.

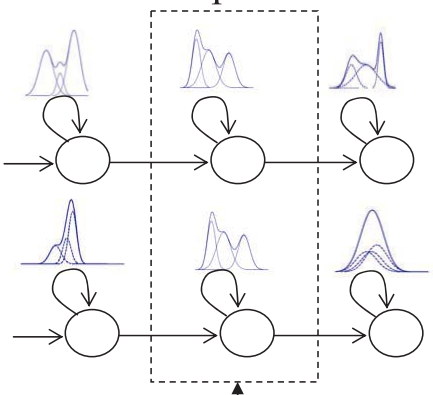
Phoneme HMMs, III

- Triphones yield accurate word models, but the trainability is reduced, 44 phonemes => 44^3=85184 different triphones.
- By sharing states between triphone HMMs, so called *tying*, trainability is improved

In theory, there are 44^2 =1936 different allophones of the phoneme /ow/. Below we have two of those:

HMM for /s **ow** l/

HMM for /k **ow** p/



It is likely that the middle state output distributions have similar parameters. Thus these can be shared between all the 1936 allophone HMMs for /ow/.

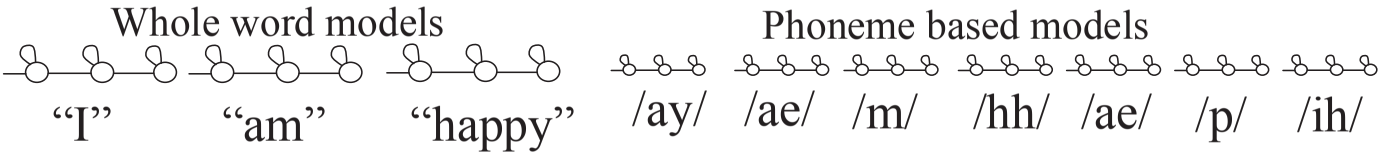
(Actually there are fewer triphones since, e.g., /ow **ow** l/ isn't a valid triphone)

Searching for the Best Word Sequence

- The MAP estimate is

$$W_* = \arg\max_W P(W \mid X) = \arg\max_W P(X \mid W)P(W)$$

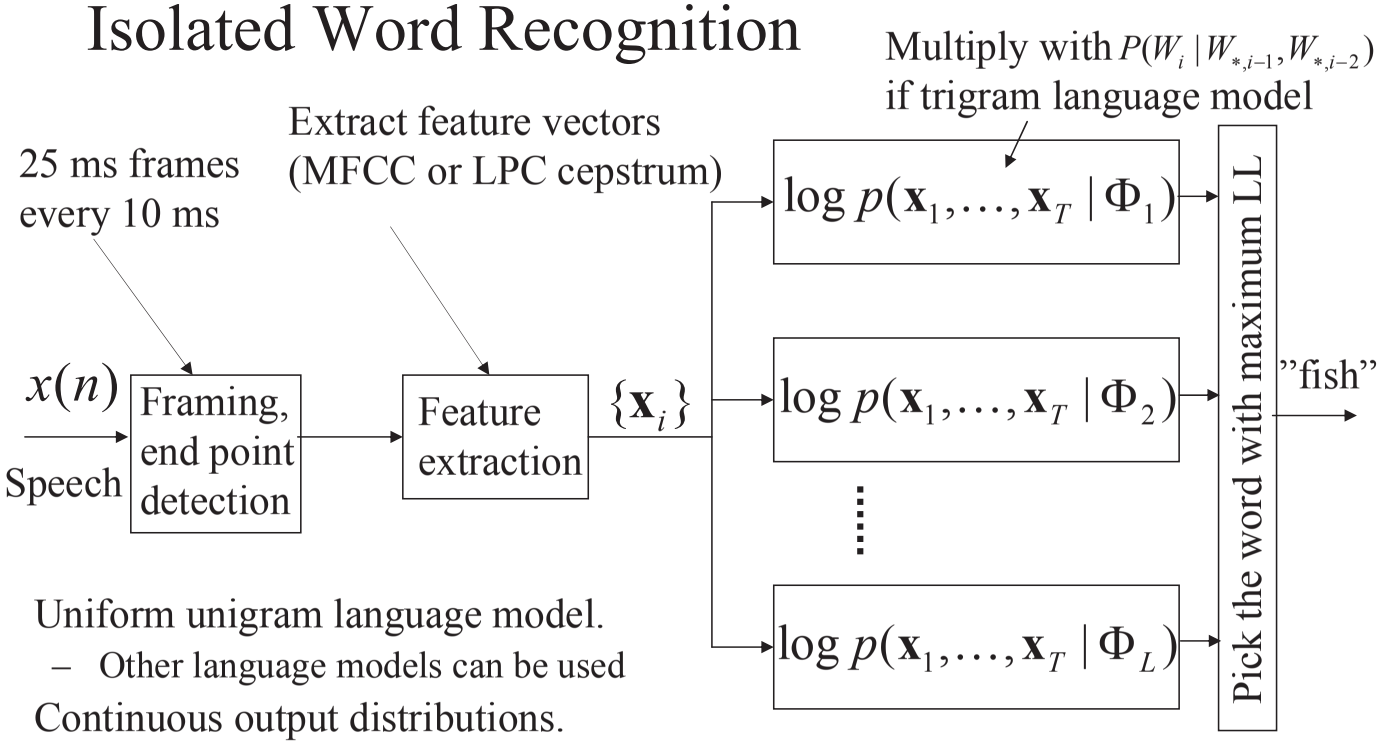
- A brute force search would be to take every possible word sequence and evaluate the acoustic model probability (using the forward algorithm), and the language model probability. Example of acoustic models for "I am happy"



- Isolated word recognition is significantly easier. The pauses between words makes it easy to detect start and end, and we can detect one word at a time like

$$W_{*,i} = \arg\max_{W_i} P(X^{(i)} \mid W_i)P(W_i \mid W_{*,i-1}, W_{*,i-2})$$

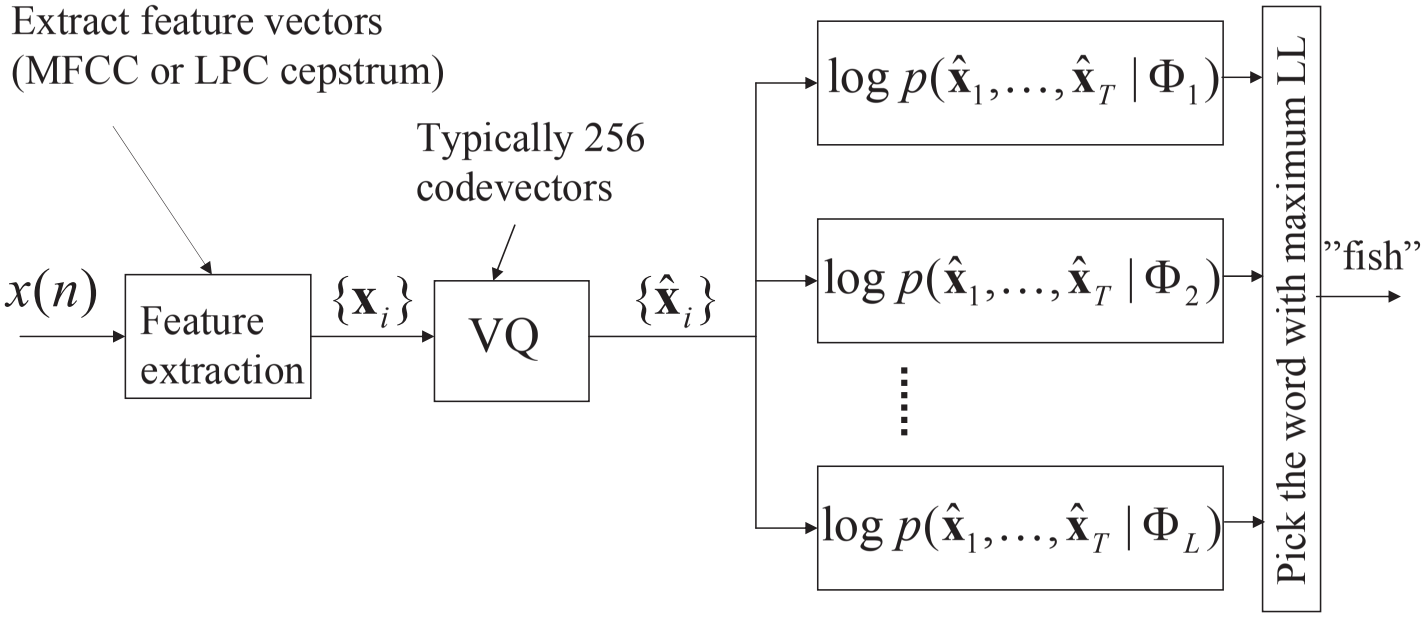
Isolated Word Recognition



- Uniform unigram language model.
  - Other language models can be used
- Continuous output distributions.
- For both whole-word HMMs , and phoneme based word models, the loglikelihoods can be evaluated using the forward algorithm.
- The Viterbi algorithm is often used since further simplifications to reduce computational complexity are possible.

$$p(\mathbf{x}_1, \ldots, \mathbf{x}_T \mid \Phi_l) \approx \max_S p(S_1, \ldots, S_T, \mathbf{x}_1, \ldots, \mathbf{x}_T \mid \Phi_l)$$

Isolated Word Recognition, Discrete HMMs

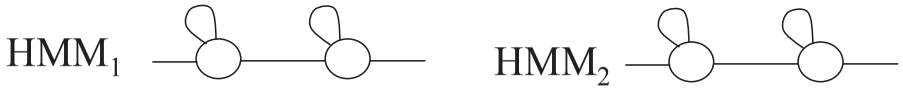


- The VQ typically has 256 codevectors.
- The output distributions are discrete; each state of the HMMs contains a list of probabilities of the VQ codevectors given that state.
- Continuous HMMs perform better than discrete HMMs, but discrete HMMs are less complex.

W1="help", W2="me"

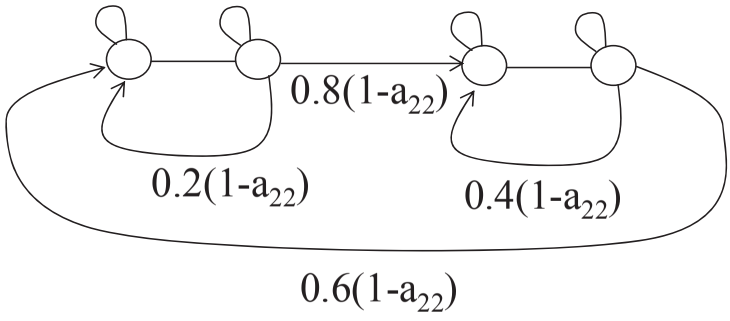
Continuous Speech Recognition I

- Cont. SR ⇔ We don't know the word boundaries in the feature sequence
- Case study: Dictonary of two words. Whole word HMMs. Bigram language model.
- The acoustic models are 2-state left-to right HMMs

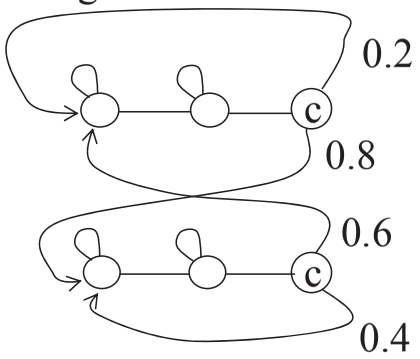


- The language model is $P(W1|W1)=0.2$, $P(W1|W2)=0.6$, $P(W2|W1)=0.8$, $P(W2|W2)=0.4$.
- The acoustic model and the language model can be combined into one HMM

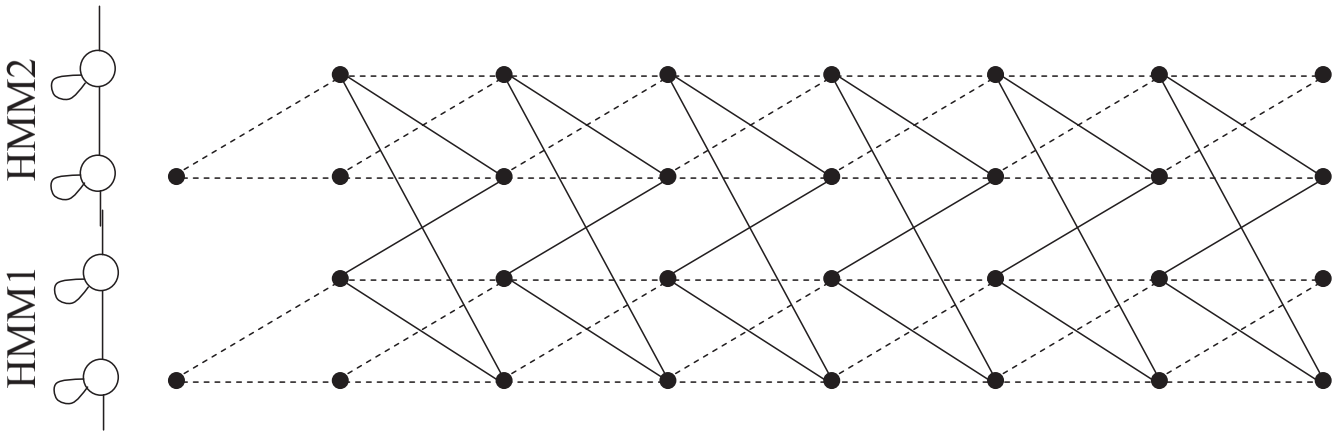
Direct connection:



Using collector states:



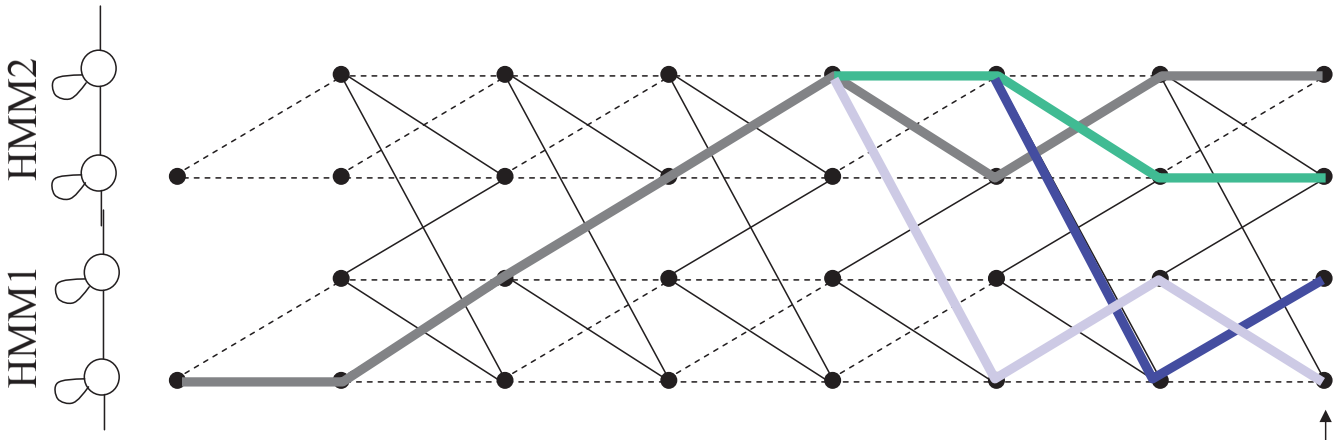
The Trellis in Our Case Study



Dashed transitions are intra-word transitions, i.e, transitions inside a word HMM.

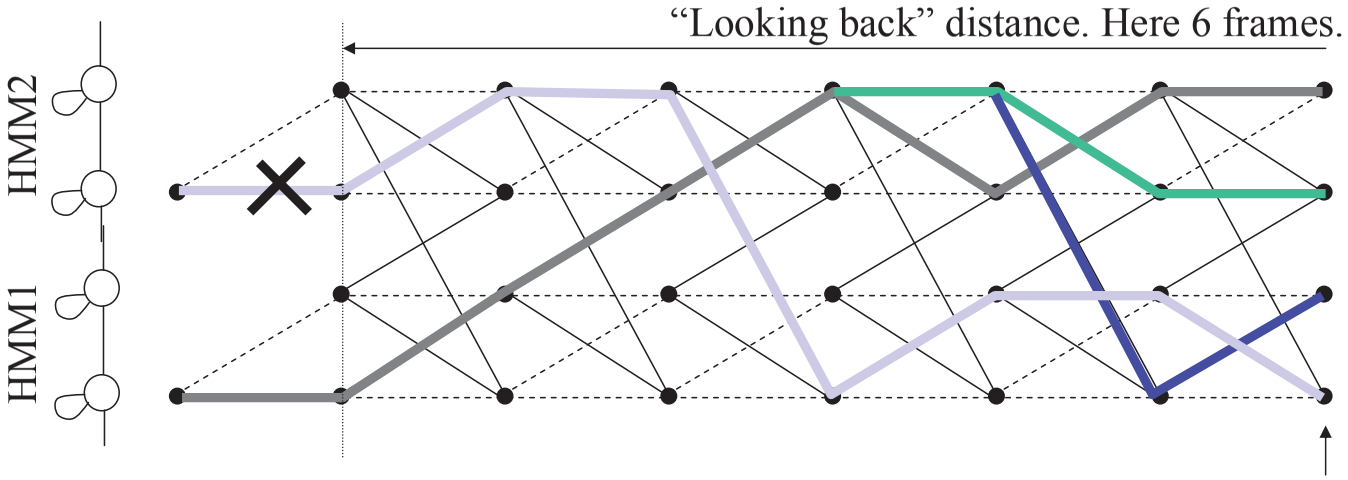
Solid transitions are inter-word transitions

Viterbi Search of The Trellis, I



- During Viterbi, the best path sequences will merge if we look back in the trellis.
- Thus, even though we have not seen the whole observation, we can make a correct decision on parts of the best path.
- If we look back far enough, we are guaranteed to have merged paths.
- To output a word from the recognizer we need to keep track of when the best path exits a word model.

Viterbi Search of The Trellis, II



- In practice, storage and computational complexity, and tolerable delay limits how far back we can look.
- With a "looking back" distance or delay *D*, we are forced at time *t* to make a decision for time *t*-*D*-1.
- The decision is based on the best-path probabilities at time *t*.
- We are *pruning* the trellis (if needed) after each decision

Continuous Speech Recognition II

- A conceptually simple but not so powerful system:
- An HMM where each state corresponds to a phoneme (around 50 phonemes in English).
- The HMM is ergodic, i.e. each state can be reached at any time with a non-zero probability.
- The best state sequence is found continuously using the Viterbi algorithm.
- A pronounciation dictionary is used to determine the best matching word sequence from the stream of phonemes from the Viterbi algorithm.