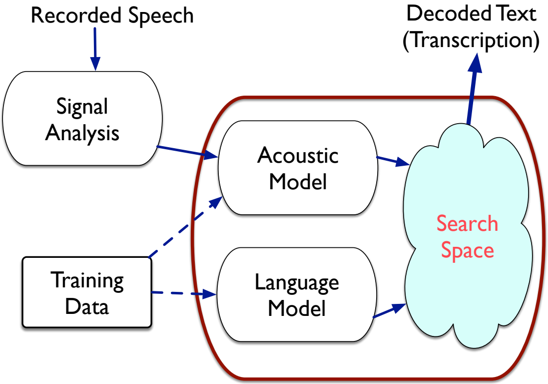
Large vocabulary ASR

Peter Bell

Automatic Speech Recognition – ASR Lecture 9
10 February 2020

HMM Speech Recognition



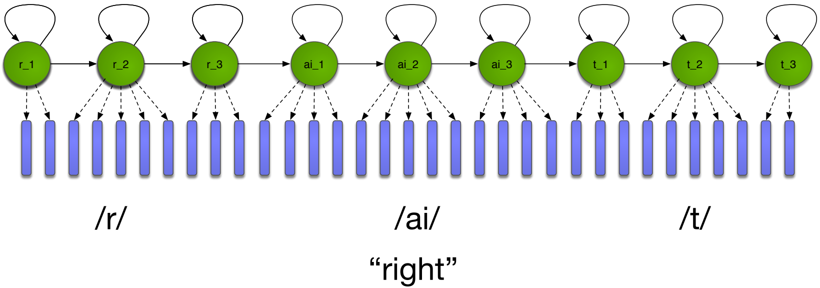
The Search Problem in ASR

- Find the most probable word sequence $\hat{W} = w_1, w_2, \ldots, w_M$ given the acoustic observations $\mathbf{X} = \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T$:

$$\begin{aligned}\hat{W} &= \arg\max_W P(W|\mathbf{X}) \\ &= \arg\max_W \underbrace{p(\mathbf{X} \mid W)}_{\text{acoustic model}} \; \underbrace{P(W)}_{\text{language model}}\end{aligned}$$

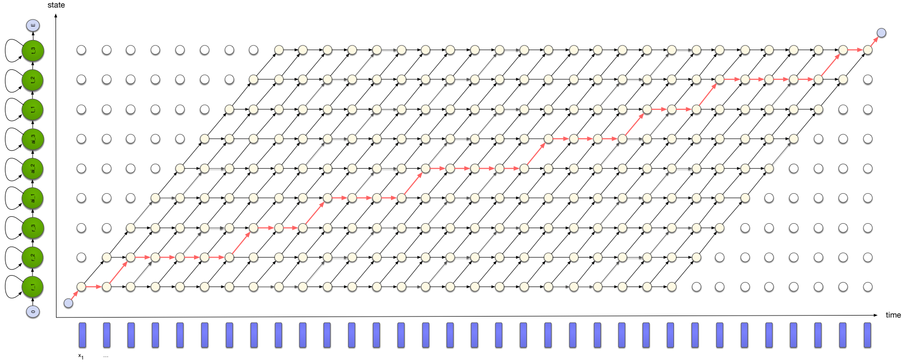
- Use pronuniciation knowledge to construct HMMs for all possible words
- Finding the most probable state sequence allows us to recover the most probable word sequence
- *Viterbi decoding* is an efficient way of finding the most probable state sequence, but even this is infeasible as the vocabulary gets very large or when a stronger language model is used

Recap: the word HMM



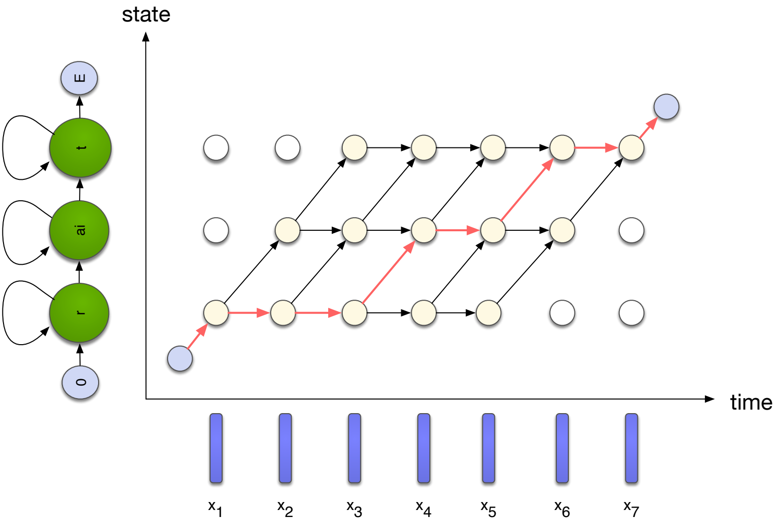
HMM naturally generates an alignment between hidden states and observation sequence

Viterbi algorithm for state alignment

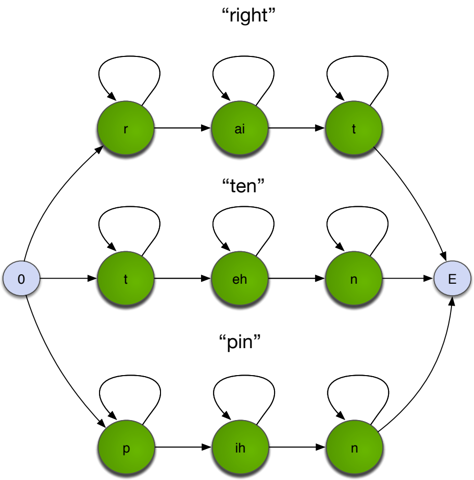


Viterbi algorithm finds the best path through the trellis – giving the highest $p(X, Q)$.

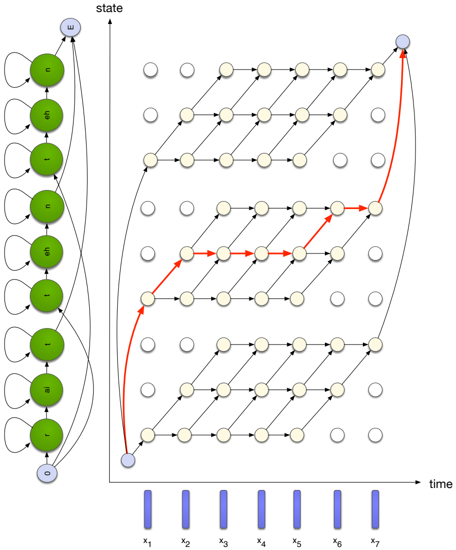
Simplified version with one state per phone



Isolated word recognition



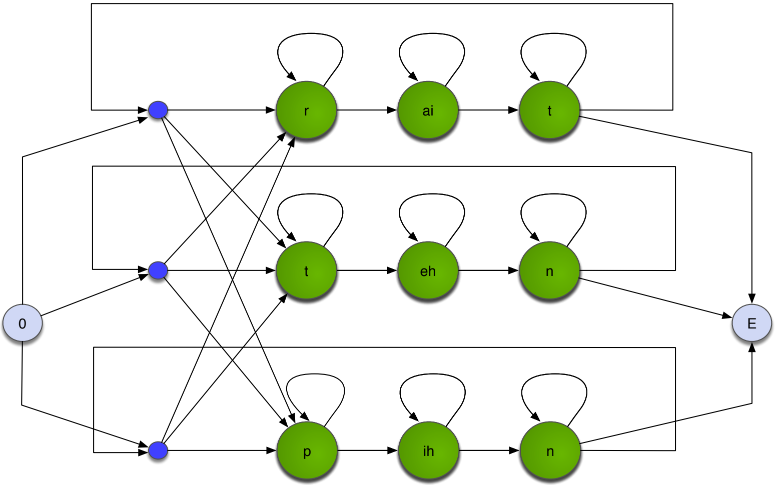
Viterbi algorithm: isolated word recognition



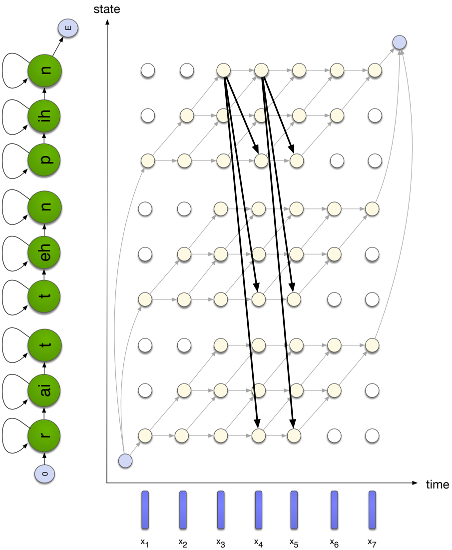
Connected word recognition

- Even worse when recognising connected words...
- The number of words in the utterance is not known
- Word boundaries are not known: any of the V words may potentially start at each frame.

Connected word recognition

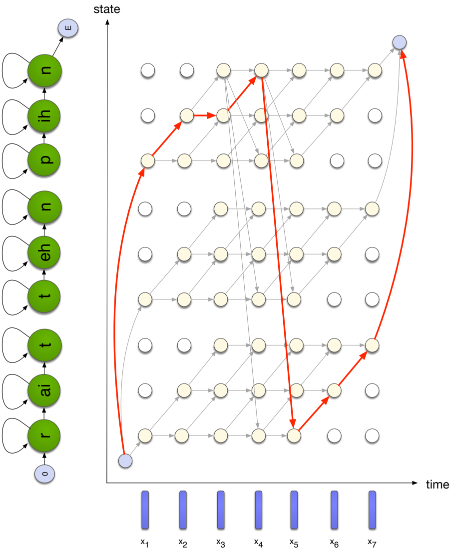


Viterbi algorithm: connected word recognition



Add transitions between all word-final and word-initial states

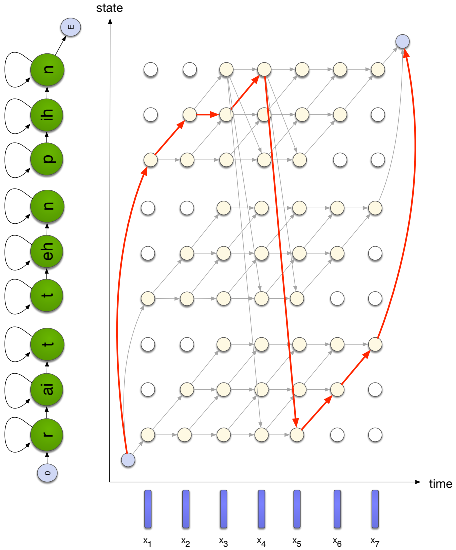
Connected word recognition



Viterbi decoding finds the best word sequence

BUT: have to consider $|V|^2$ inter-word transitions at every time step

Connected word recognition



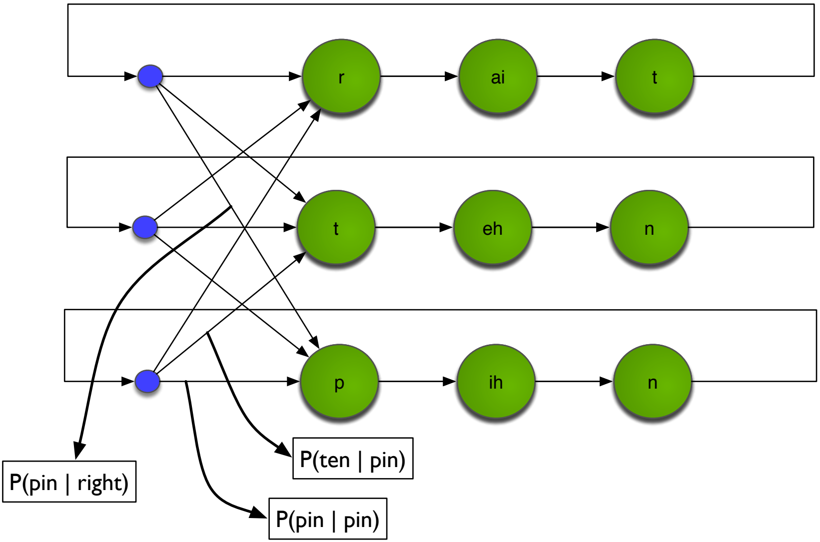
Integrating the language model

- So far we've estimated HMM transition probabilities from audio data, as part of the acoustic model
- Transitions *between words rightarrow* use a language model
- *n*-gram language model:

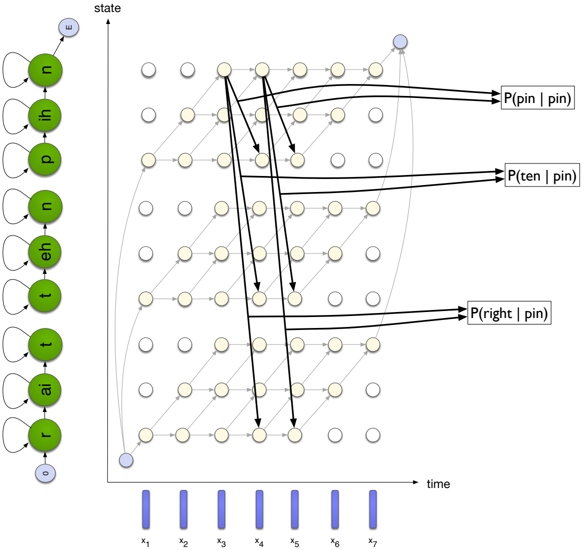
$$p(w_i|h_i) = p(w_i|w_{i-n}, \ldots w_{i-1})$$

- Integrate the language model directly in the Viterbi search

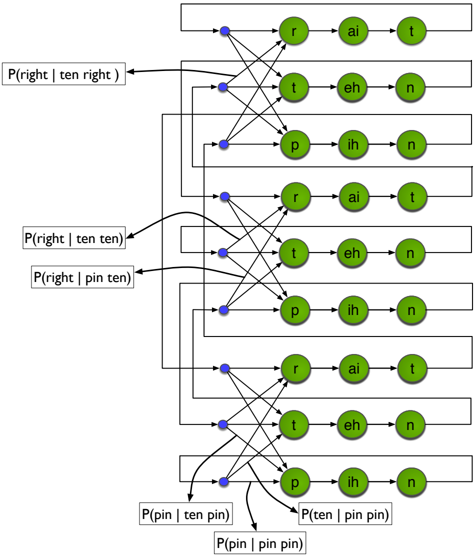
Incorporating a bigram language model



Incorporating a bigram language model



Incorporating a trigram language model



Need to duplicate HMM states to incorporate extended word history

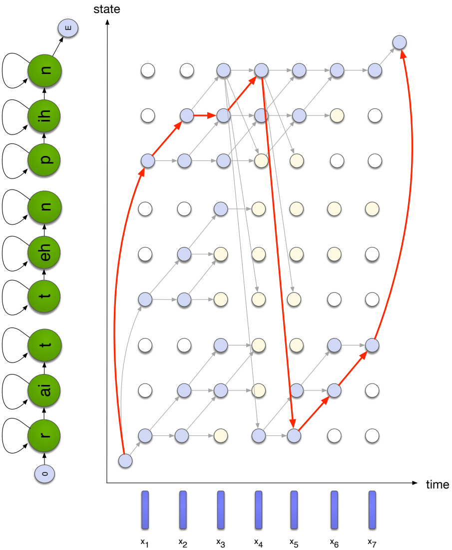
Computational Issues

- Viterbi decoding performs an exact search in an efficient manner
- But exact search is not possible for large vocabulary tasks
  - Long-span language models and the use of cross-word triphones greatly increase the size of the search space
- Solutions:
  - Beam search (prune low probability hypotheses)
  - Tree structured lexicons
  - Language model look-ahead
  - Dynamic search structures
  - Multipass search ($\rightarrow$ two-stage decoding)
  - Best-first search ($\rightarrow$ stack decoding / A^* search)

Computational Issues

- Viterbi decoding performs an exact search in an efficient manner
- But exact search is not possible for large vocabulary tasks
  - Long-span language models and the use of cross-word triphones greatly increase the size of the search space
- Solutions:
  - Beam search (prune low probability hypotheses)
  - Tree structured lexicons
  - Language model look-ahead
  - Dynamic search structures
  - Multipass search ($\rightarrow$ two-stage decoding)
  - Best-first search ($\rightarrow$ stack decoding / A^* search)
- An alternative approach: Weighted Finite State Transducers (WFST)

Pruning



During Viterbi decoding, don't propagate tokens whose probability falls a certain amount below the current best path

Result is only an approximation to the best path

Tree-structured lexicon

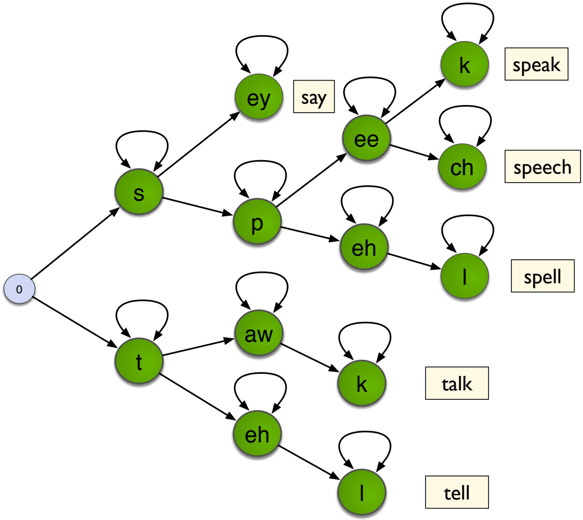
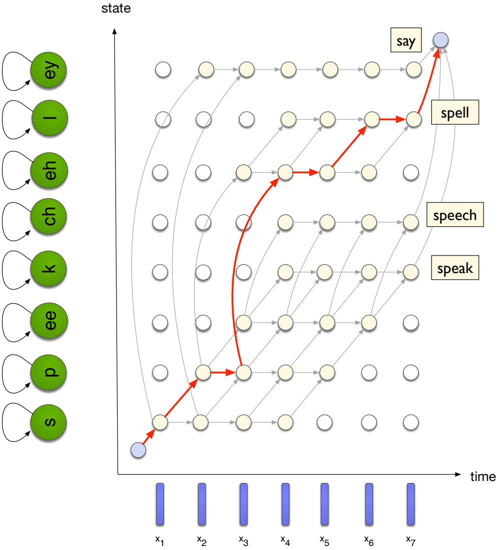


Figure adapted from Ortmans & Ney, "The time-conditioned approach in dynamic programming search for LVCSR"

Tree-structured lexicon



Reduces the number of state transition computations

For clarity, not all the connections are shown

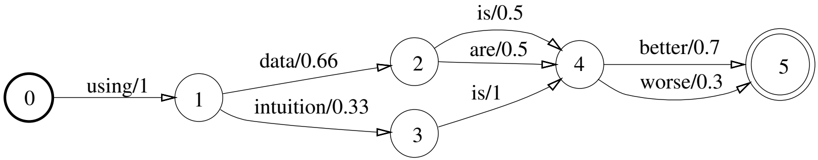
Language model look-ahead

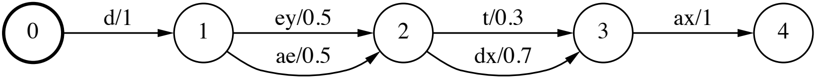
- Aim to make pruning more efficient
- In tree-structured decoding, look ahead to find out the best LM score for any words further down the tree
- This information can be pre-computed and stored at each node in the tree
- States in the tree are pruned early if we know that none of the possibilities will receive good enough probabilities from the LM.

Weighted Finite State Transducers

- Weighted finite state automaton that transduces an input sequence to an output sequence (Mohri et al 2008)
- States connected by transitions. Each transition has
  - input label
  - output label
  - weight
- Weights use the *log semi-ring* or *tropical semi-ring* – with operations that correspond to multiplication and addition of probabilities
- There is a single start state. Any state can optionally be a final state (with a weight)
- Used by Kaldi

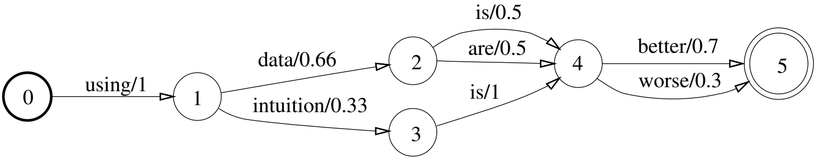
Weighted Finite State Acceptors



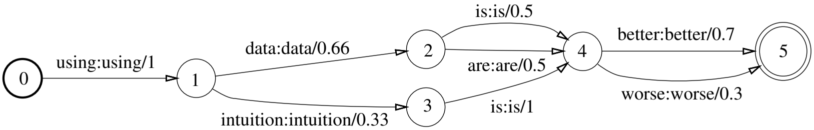


Weighted Finite State Transducers

Acceptor

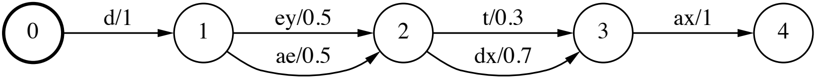


Transducer

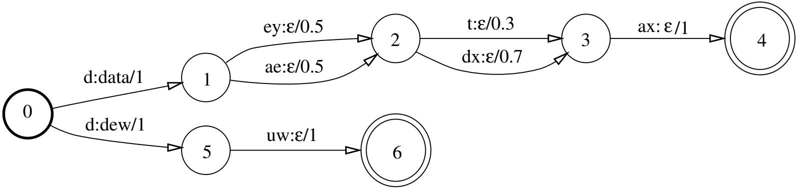


Weighted Finite State Transducers

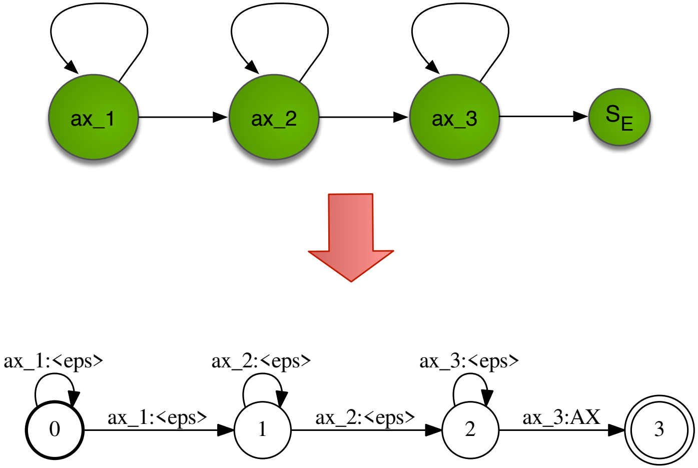
Acceptor



Transducer



The HMM as a WFST



WFST Algorithms

Composition Combine transducers T_1 and T_2 into a single transducer acting as if the output of T_1 was passed into T_2.

Determinisation Ensure that each state has no more than a single output transition for a given input label

Minimisation transforms a transducer to an equivalent transducer with the fewest possible states and transitions

Applying WFSTs to speech recognition

- Represent the following components as WFSTs

	transducer	input sequence	output sequence
G	word-level grammar	words	words
L	pronunciation lexicon	phones	words
C	context-dependency	CD phones	phones
H	HMM	HMM states	CD phones

- Composing L and G results in a transducer $L \circ G$ that maps a phone sequence to a word sequence
- $H \circ C \circ L \circ G$ results in a transducer that maps from HMM states to a word sequence

Reading

- Ortmanns and Ney (2000). "The time-conditioned approach in dynamic programming search for LVCSR". In IEEE Transactions on Speech and Audio Processing, Vol. 8, No. 6.
- Mohri et al (2008). "Speech recognition with weighted finite-state transducers." In Springer Handbook of Speech Processing, pp. 559-584. Springer.
  http://www.cs.nyu.edu/~mohri/pub/hbka.pdf
- WFSTs in Kaldi. http://danielpovey.com/files/Lecture4.pdf