

# Search and Decoding

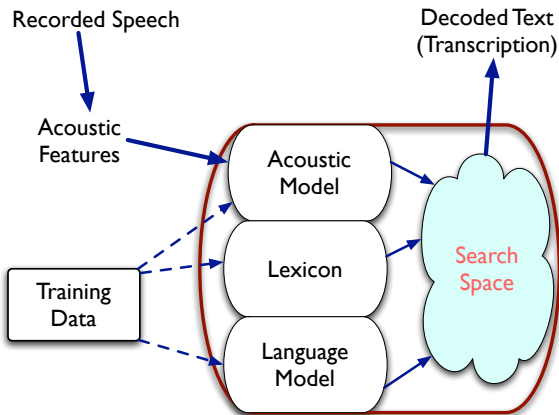
Steve Renals

Automatic Speech Recognition— ASR Lecture 10  
23 February 2009

## Today's lecture

- Search in (large vocabulary) speech recognition
- Viterbi decoding
- Approximate search

# HMM Speech Recognition



# The Search Problem in ASR (1)

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

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

- Words are composed of state sequences so we may express this criterion by summing over all state sequences

$$Q = q_1, q_2, \dots, q_n:$$

$$\hat{W} = \arg \max_W P(W) \sum_Q P(Q | W) P(\mathbf{X} | Q)$$

The acoustic observation sequence is conditionally independent of the word sequence given the HMM state sequence.

# The Search Problem in ASR (2)

- **Viterbi criterion:** approximate the sum over all state sequences by using the most probable state sequence:

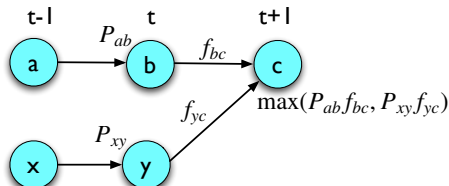
$$\hat{W} = \arg \max_W P(W) \max_{Q \in \mathcal{Q}_W} P(Q | W) P(X | Q)$$

$\mathcal{Q}_W$  is the set of all state sequences corresponding to word sequence  $W$

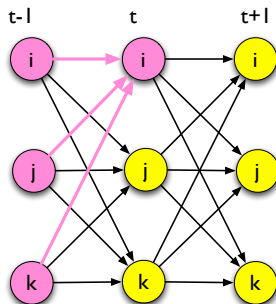
- The task of the search (or decoding) algorithm is to determine  $\hat{W}$  using the above equation given the acoustic, pronunciation and language models
- In a large vocabulary task evaluating all possible word sequences is infeasible (even using an efficient exact algorithm)
  - Reduce the size of the search space through pruning unlikely hypotheses
  - Eliminate repeated computations

# Viterbi Decoding

- Naive exhaustive search: with a vocabulary size  $V$ , and a sequence of  $M$  words, there are  $V^M$  different alternatives to consider!
- Viterbi decoding (forward dynamic programming) is an efficient, recursive algorithm that performs an optimal exhaustive search
- For HMM-based speech recognition, the Viterbi algorithm is used to find the most probable path through a probabilistically scored time/state lattice
- Exploits first-order Markov property—only need to keep the most probable path at each state:

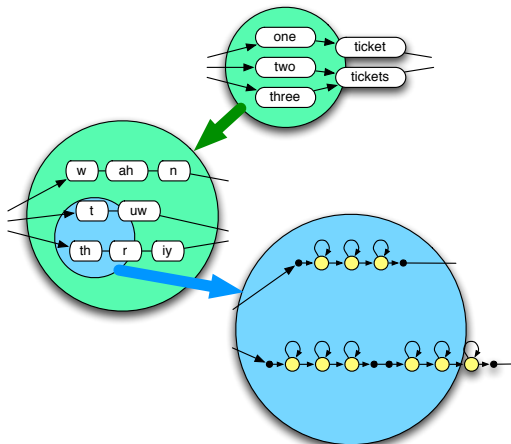


# Time-state trellis



- Set up the problem as a trellis of states and times
- Use the Viterbi approximation
- At each state-time point keep the single most probable path, discard the rest
- The most probable path is the one at the end state at the final time
- Typically use log probabilities

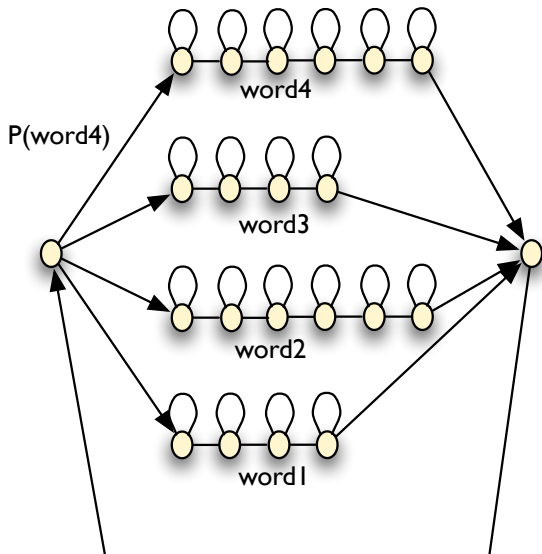
# Compiling a Recognition Network



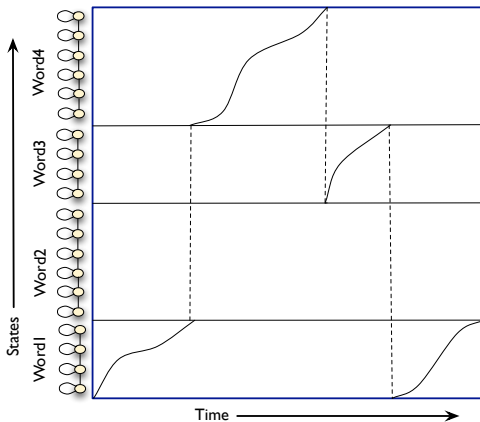
Build a network of HMM states from a network of phones from a network of words



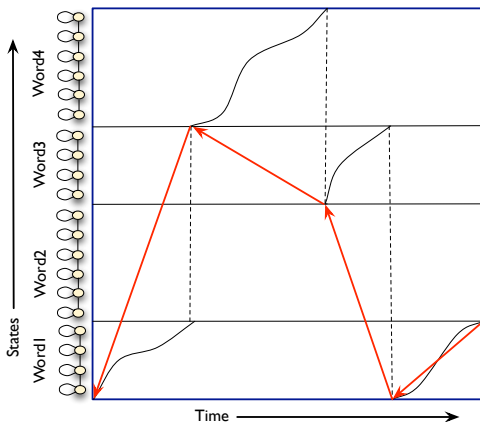
# Connected Word Recognition



# Time Alignment Path

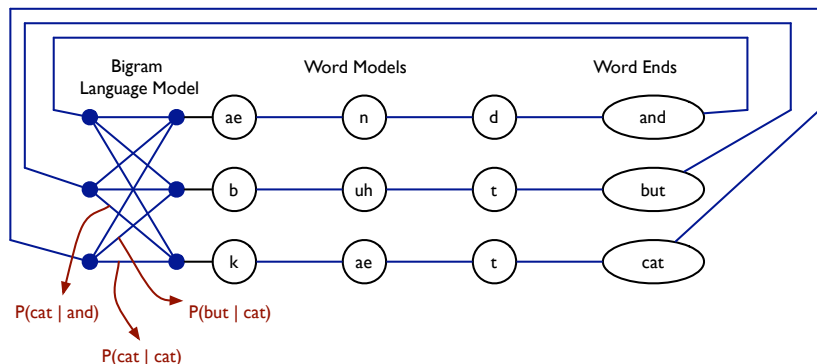


# Backtrace to Obtain Word Sequence



- Backpointer array keeps track of word sequence for a path:  
 $\text{backpointer}[\text{word}][\text{wordStartFrame}] = (\text{prevWord}, \text{prevWordStartFrame})$
- Backtrace through backpointer array to obtain the word sequence for a path

# Incorporating a bigram language model

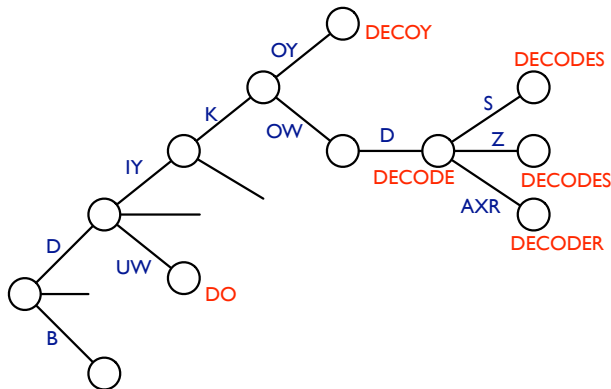


Trigram or longer span models require a word history.

- Viterbi decoding performs an exact search in an efficient manner
- Exact search is not possible for large vocabulary tasks. If the vocab size is  $V$ :
  - Word boundaries are not known:  $V$  words may potentially start at each frame
  - Cross-word triphones need to be handled carefully since the acoustic score of a word-final phone depends on the initial phone of the next word
  - Long-span language models (eg trigrams) greatly increase the size of the search space
- Solutions:
  - Beam search (prune low probability hypotheses)
  - Dynamic search structures
  - Multipass search
  - Best-first search
  - Finite State Transducer (FST) approaches

# Sharing Computation: Prefix Pronunciation Tree

- Need to build an HMM for each word in the vocabulary
- Individual HMM for each word results in phone models duplicated in different words
- Share computation by arranging the lexicon as a tree



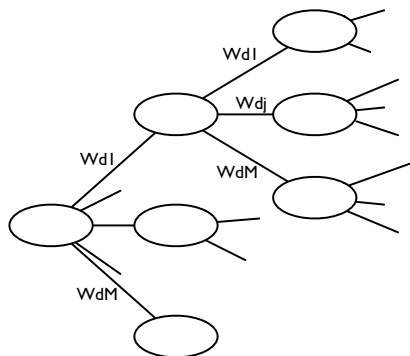
- **Basic idea:** Prune search paths which are unlikely to succeed
- Remove nodes in the time-state trellis whose path probability is more than a factor  $\delta$  less probable than the best path (only consider paths in the beam)
- Both language model and acoustic model can contribute to pruning
- Pronunciation tree can limit pruning since the language model probabilities are only known at word ends: each internal node can keep a list of words it contributes to
- Search errors: errors arising due to the fact that the most probable hypothesis was incorrectly pruned
- Need to balance search errors with speed

# Multipass Search

- Rather than compute the single best hypothesis the decoder can output alternative hypotheses
- $N$ -best list: list of the  $N$  most probable hypotheses
- Word Graph/Word Lattice:
  - Nodes correspond to time (frame)
  - Arcs correspond to word hypotheses (with associated acoustic and language model probabilities)
- Multipass search using progressively more detailed models
  - Eg: use bigram language model on first pass, trigram on second pass
  - Transmit information between passes as word graphs
  - Later passes rescore word graphs produced by earlier passes



# Word Search Tree



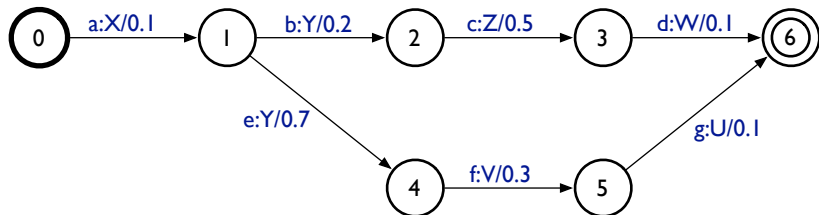
- View recognition search as searching a tree
- Viterbi decoding is breadth-first search — **time-synchronous**
- Pruning deactivates part of the search tree
- Also possible to use best first search (stack decoding) — **time asynchronous**

# Static and dynamic networks

- Previous approaches constructed the search space *dynamically*: less probable paths are not explored.
- Dynamic search is resource-efficient but results in
  - complex software
  - tight interactions between pruning algorithms and data structures
- Static networks are efficient for smaller vocabularies, but not immediately applicable to large vocabularies
- Efficient static networks would enable
  - Application of network optimization algorithms in advance
  - Decoupling of search network construction and decoding

# Weighted Finite State Transducers

- Finite state automaton that transduces an input sequence to an output sequence
- States connected by transitions. Each transition has
  - input label
  - output label
  - weight



**Composition** Used to combine transducers at different levels. For example if  $G$  is a finite state grammar and  $P$  is a pronunciation dictionary then  $D$  transduces a phone string to any word string, whereas  $P \circ G$  transduces a phone string to word strings allowed by the grammar

**Determinisation** removes non-determinacy from the network by ensuring 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

Several libraries for WFSTs eg:

- Open FST: <http://www.openfst.org/>
- MIT: <http://people.csail.mit.edu/ilh/fst/>
- AT&T: <http://www.research.att.com/~fsmtools/fsm/>

# WFST-based decoding

- Represent the following components as WFSTs
  - Context-dependent acoustic models ( $C$ )
  - Pronunciation dictionary ( $D$ )
  - $n$ -gram language model ( $L$ )
- The decoding network is defined by their composition:  
 $C \circ D \circ L$
- Successively determinize and combine the component transducers, then minimize the final network
- Problem: although the final network may be of manageable size, the construction process may be very memory intensive, particularly with 4-gram language models or vocabularies of over 50,000 words
- Used successfully in several systems

- How accurate is a speech recognizer?
- Use dynamic programming to align the ASR output with a reference transcription
- Three type of error: insertion, deletion, substitution
- Word error rate (WER) sums the three types of error. If there are  $N$  words in the reference transcript, and the ASR output has  $S$  substitutions,  $D$  deletions and  $I$  insertions, then:

$$\text{WER} = 100 \cdot \frac{S + D + I}{N} \% \quad \text{Accuracy} = 100 - \text{WER} \%$$

- Speech recognition evaluations: common training and development data, release of new test sets on which different systems may be evaluated using word error rate
  - NIST evaluations enabled an objective assessment of ASR research, leading to consistent improvements in accuracy
  - May have encouraged incremental approaches at the cost of subduing innovation (“Towards increasing speech recognition error rates”)

- Search in speech recognition
- Viterbi decoding
- Connected word recognition
- Incorporating the language model
- Pruning
- Prefix pronunciation trees
- Weighted finite state transducers
- Evaluation

- Aubert (2002) - review of decoding techniques
- Mohri et al (2002) - WFSTs applied to speech recognition
- Moore et al (2006) - Juicer (example of a WFST-based decoder)