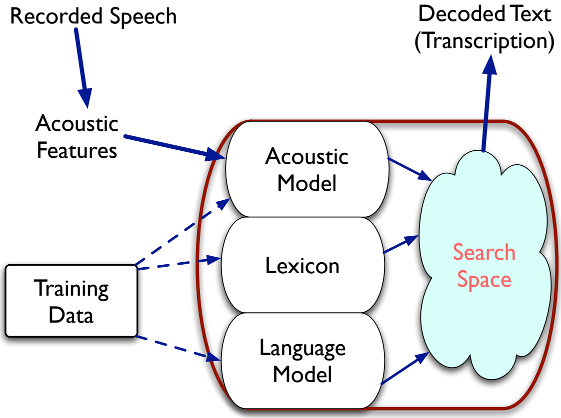
Decoding and WFSTs

Steve Renals

Automatic Speech Recognition – ASR Lecture 13
9 March 2017

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 $\mathbf{X} = \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$:

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

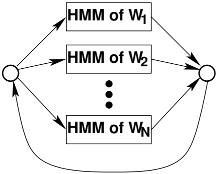
- Words are composed of state sequences so this problem corresponds to finding the most probable allowable state sequence (given the constraints of pronunciation lexicon and language model) - **Viterbi decoding**
- In a large vocabulary task evaluating all possible word sequences in infeasible (even using an efficient exact algorithm)
  - Reduce the size of the search space through pruning unlikely hypotheses
  - Eliminate repeated computations

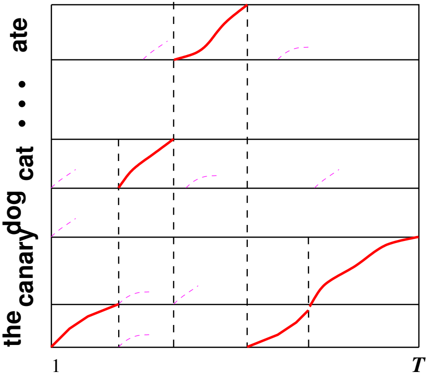
Connected Word Recognition

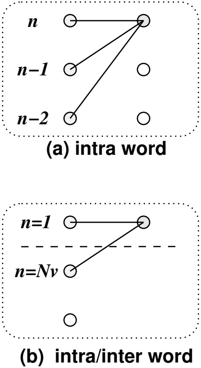
- The number of words in the utterance is not known
- Word boundaries are not known: V words may potentially start at each frame

Connected Word Recognition

- The number of words in the utterance is not known
- Word boundaries are not known: V words may potentially start at each frame

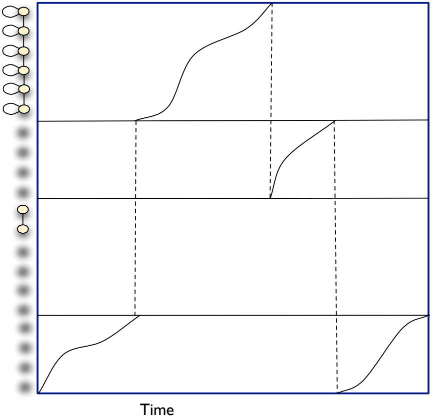




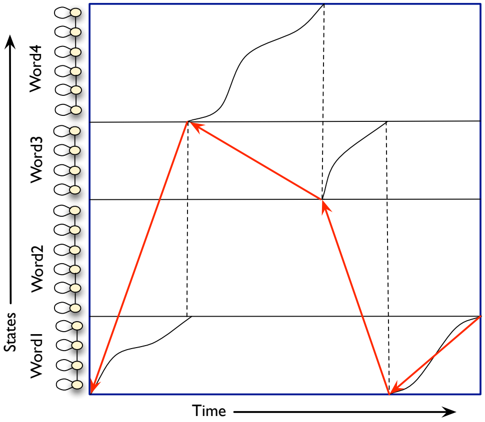


speech: "the cat ate the canary"

Time Alignment Path

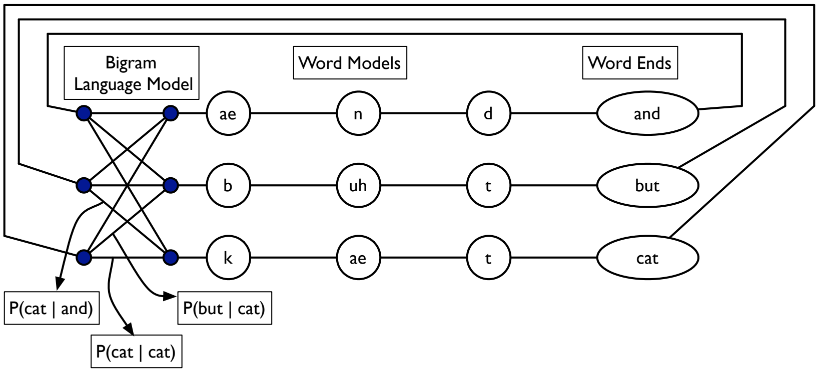


Backtrace to Obtain Word Sequence



- Backpointer array keeps track of word sequence for a path:
  backpointer[word][wordStartFrame] = (prevWord, 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.

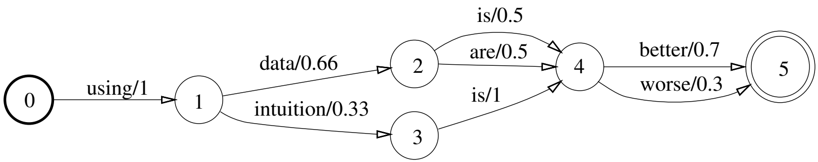
Computational Issues

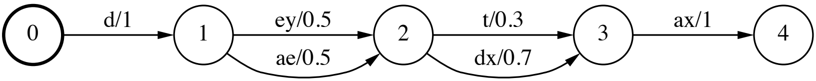
- Viterbi decoding performs an exact search in an efficient manner
- Exact search is not possible for large vocabulary tasks
  - Cross-word triphones need to be handled carefully since the acoustic score of a word-final phone depends on the initial phone of the next word
  - Long-span language models (eg trigrams) greatly increase the size of the search space
- Solutions:
  - Beam search (prune low probability hypotheses)
  - Dynamic search structures
  - Multipass search ($\rightarrow$ two-stage decoding)
  - Best-first search ($\rightarrow$ stack decoding / A^* search)
  - An alternative approach: Weighted Finite State Transducers (WFST)

Weighted Finite State Transducers

- Used by Kaldi
- Weighted finite state automaton that transduces an input sequence to an output sequence (Mohri 2008)
- States connected by transitions. Each transition has
  - input label
  - output label
  - weight

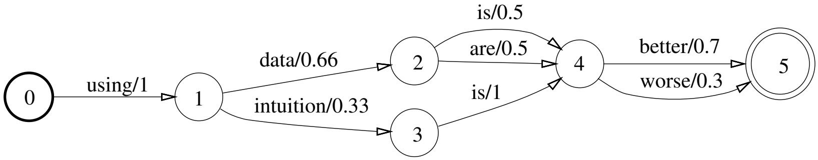
Weighted Finite State Acceptors



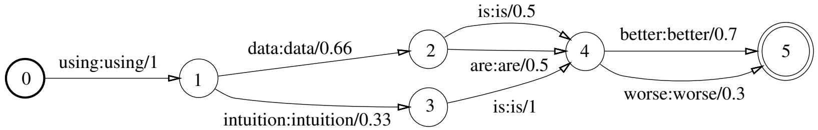


Weighted Finite State Transducers

Acceptor

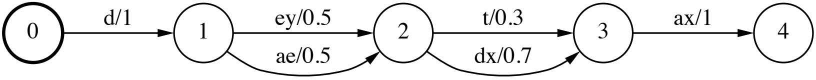


Transducer

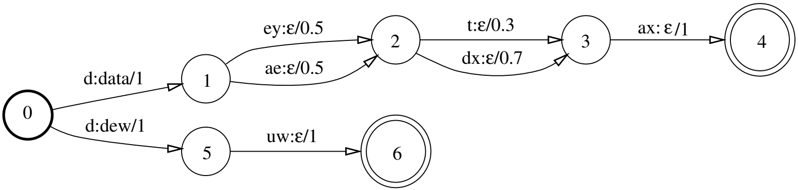


Weighted Finite State Transducers

Acceptor



Transducer



WFST Algorithms

Composition Combine transducers at different levels. For example if G is a finite state grammar and L is a pronunciation dictionary then $L \circ G$ transduces a phone string to word strings allowed by the grammar

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

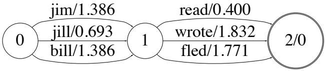
- Represent the following components as WFSTs

	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

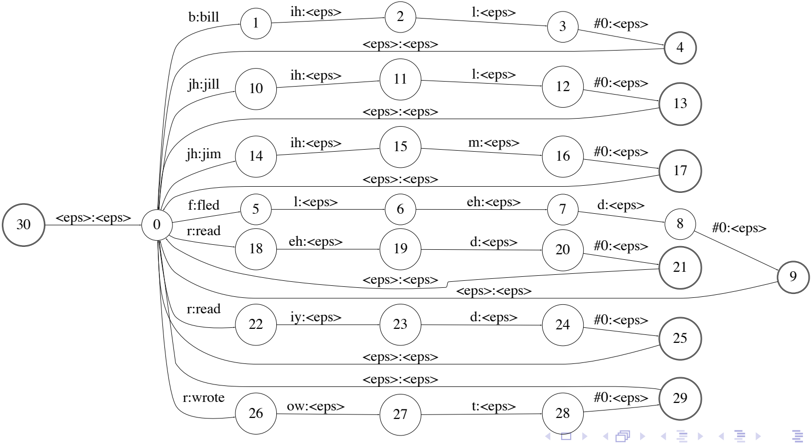
L, G

G



jim/1.386
jill/0.693
bill/1.386
read/0.400
wrote/1.832
fled/1.771
0
1
2/0

L

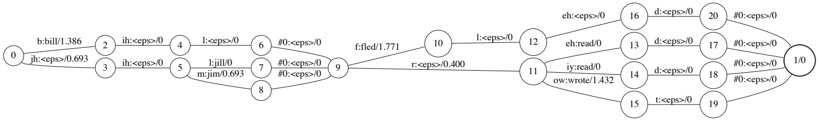


30
<eps>:<eps>
0
b:bill
1
ih:<eps>
2
l:<eps>
3
#0:<eps>
4
<eps>:<eps>
jh:jill
10
ih:<eps>
11
l:<eps>
12
#0:<eps>
13
<eps>:<eps>
jh:jim
14
ih:<eps>
15
m:<eps>
16
#0:<eps>
17
<eps>:<eps>
f:fled
5
l:<eps>
6
eh:<eps>
7
d:<eps>
8
#0:<eps>
9
r:read
18
eh:<eps>
19
d:<eps>
20
#0:<eps>
21
<eps>:<eps>
<eps>:<eps>
r:read
22
iy:<eps>
23
d:<eps>
24
#0:<eps>
25
<eps>:<eps>
<eps>:<eps>
r:wrote
26
ow:<eps>
27
t:<eps>
28
#0:<eps>
29

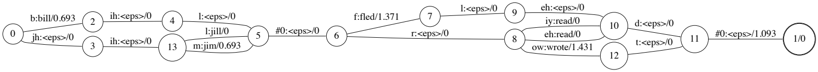
$L \circ G$, $\det(L \circ G)$, $\min(\det(L \circ G))$

$L \circ G$

$\det(L \circ G)$

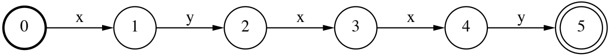


$\min(\det(L \circ G))$

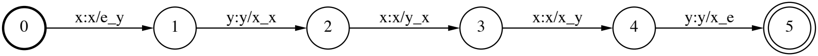


Context dependency transducer C

Context-independent "string"



Context-dependency transducer (weights not shown)



(`x/e_y` – `x` with left context `e` (start/end) and right context `y`)

Decoding using WFSTs

- We can represent the HMM acoustic model, pronunciation lexicon and n-gram language model as four transducers: H, C, L, G
- Combining the transducers gives an overall “decoding graph” for our ASR system – but minimisation and determination means it is much smaller than naively combining the transducers
- But it is important in which order the algorithms are combined otherwise the transducers may “blow-up” – basically after each composition, first determinise then minimise
- In Kaldi, ignoring one or two details

$$HCLG = \min(\det(H \circ \min(\det(C \circ \min(\det(L \circ G))))))$$

Reading

- Mohri (2008) – Mohri, Pereira, and Riley (2008). "Speech recognition with weighted finite-state transducers." In Springer Handbook of Speech Processing, pp. 559-584. Springer, 2008.
  http://www.cs.nyu.edu/~mohri/pub/hbka.pdf
- Decoding and WFSTs in Kaldi –
  http://danielpovey.com/files/Lecture4.pdf