Lecture 9

LVCSR Search

Michael Picheny, Bhuvana Ramabhadran, Stanley F. Chen, Markus Nussbaum-Thom

Watson Group
IBM T.J. Watson Research Center
Yorktown Heights, New York, USA
{picheny,bhuvana,stanchen,nussbaum}@us.ibm.com

23 March 2016

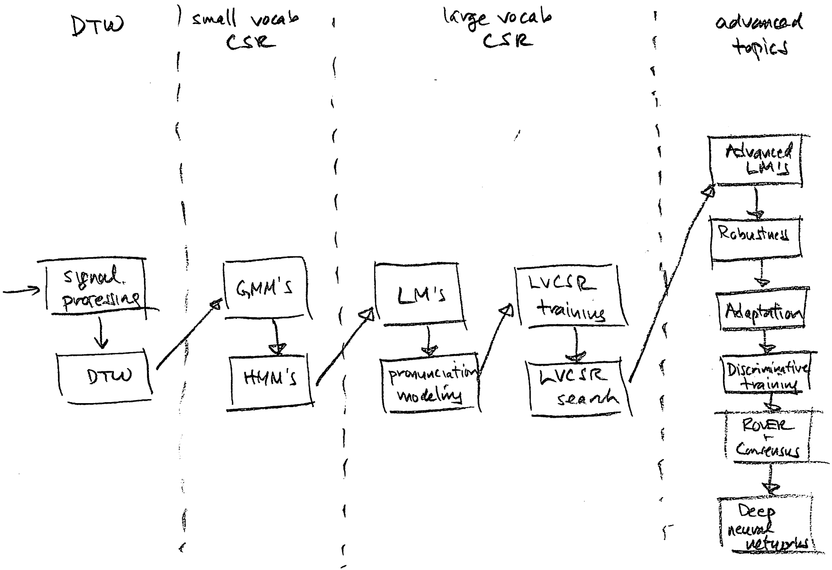
Administrivia

- Lab 2 sample answers.
  - `/user1/faculty/stanchen/e6870/lab2_ans/`
- Lab 3 not graded yet.
- Lab 4 out today.
  - Due nine days from now (Friday, Apr. 1) at 6pm?
- Lab 5 cancelled.
- Visit to IBM Watson Astor Place in 1.5 weeks.
  - April 1, 11am-1pm.

Feedback

- Clear (2); mostly clear (1).
- Pace: fast (1).
- Muddiest: moving from small to large vocab (1).
- No comments with 2+ votes; 6 responses total.

Road Map



DTW
small vocab CSR
large vocab CSR
advanced topics
signal processing
DTW
GMM's
HMM's
LM's
pronunciation modeling
LVCSR training
LVCSR search
Advanced LM's
Robustness
Adaptation
Discriminative training
ROVER + Consensus
Deep neural networks

Review, Part I

- What is $\mathbf{x}$?
  - The feature vector.
- What is ω?
  - A word sequence.
- What notation do we use for acoustic models?
  - $P(\mathbf{x}|\omega)$
- What does an acoustic model model?
  - How likely feature vectors are given a word sequence.
- What notation do we use for language models?
  - $P(\omega)$
- What does a language model model?
  - How frequent each word sequence is.

Review, Part II

- What is the fundamental equation of ASR?

$$
\begin{aligned}
\text{(answer)} &= \operatorname*{arg\,max}_{\omega \in \text{vocab}^*} \text{(language model)} \times \text{(acoustic model)} \\
&= \operatorname*{arg\,max}_{\omega \in \text{vocab}^*} \text{(prior prob over words)} \times P(\text{feats}|\text{words}) \\
&= \operatorname*{arg\,max}_{\omega \in \text{vocab}^*} P(\omega) P(\mathbf{x}|\omega)
\end{aligned}
$$

Match the Lecture With The Topic

Language modeling	Estimate $P(\mathbf{x}\|\omega)$
LVCSR training	$\arg\max_{\omega \in \text{vocab}^*} P(\omega)P(\mathbf{x}\|\omega)$
LVCSR search	Estimate $P(\omega)$

- Which of these are offline? Online?

Demo: Speed Kills

This Lecture

- How to do LVCSR decoding.
- How to make it fast.

Part I

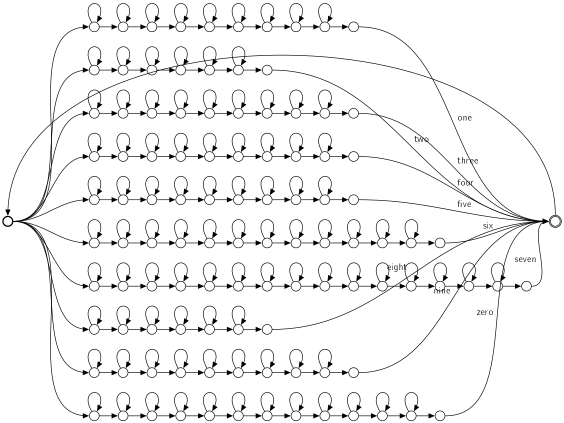
Making the Decoding Graph

LVCSR Search a.k.a. Decoding

$$
\begin{aligned}
\text{(answer)} &= \operatorname*{arg\,max}_{\omega \in \text{vocab}^*} \text{(language model)} \times \text{(acoustic model)} \\
&= \operatorname*{arg\,max}_{\omega \in \text{vocab}^*} P(\omega) P(\mathbf{x}|\omega)
\end{aligned}
$$

- How to compute the argmax?
  - Run Viterbi/Forward/Forward-Backward?
  - One big HMM/one small HMM/lots of small HMM's?
- The whole ballgame: how to build the HMM!!!

One Big HMM: Small Vocabulary



Small $\Rightarrow$ Large Vocabulary

- How to build the big HMM for LVCSR?
- What's missing? Are there any scores we need to add?

Idea: Add LM Scores to HMM

$$\begin{aligned}(\text{answer}) &= \operatorname*{arg\,max}_{\omega \in \text{vocab}^*} (\text{language model}) \times (\text{acoustic model}) \\ &= \operatorname*{arg\,max}_{\omega \in \text{vocab}^*} P(\omega)P(\mathbf{x}|\omega)\end{aligned}$$

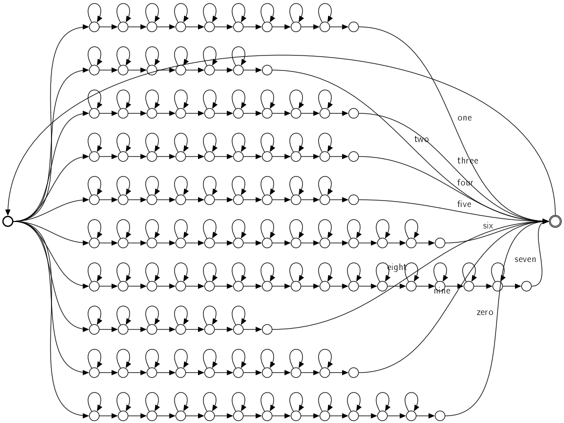
- Viterbi: without LM.

$$\operatorname*{arg\,max}_{\omega} P(\mathbf{x}|\omega) \Leftrightarrow \max \prod_{t=1}^{T} (\text{arc cost})$$

- Viterbi: with LM.

$$\operatorname*{arg\,max}_{\omega} P(\omega)P(\mathbf{x}|\omega) \Leftrightarrow \operatorname*{arg\,max} \prod_{t=1}^{T} (\text{arc cost}) \times (\text{LM score})$$

Adding in Unigram LM Scores $P(w_i)$



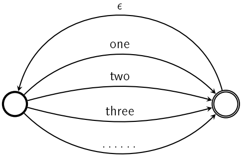
- What about bigram $P(w_i|w_{i-1})$? Trigrams $P(w_i|w_{i-2}w_{i-1})$?

Adding Language Model Scores

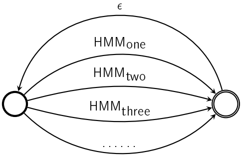
- Solution: multiple copies of each word HMM!
- Old view: add LM scores to word HMM loop.
- New view: express LM as HMM. Sub in word HMM's.

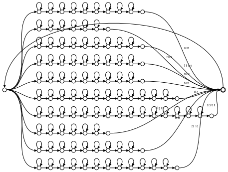
Example: Unigram LM

- Take (H)MM representing language model.

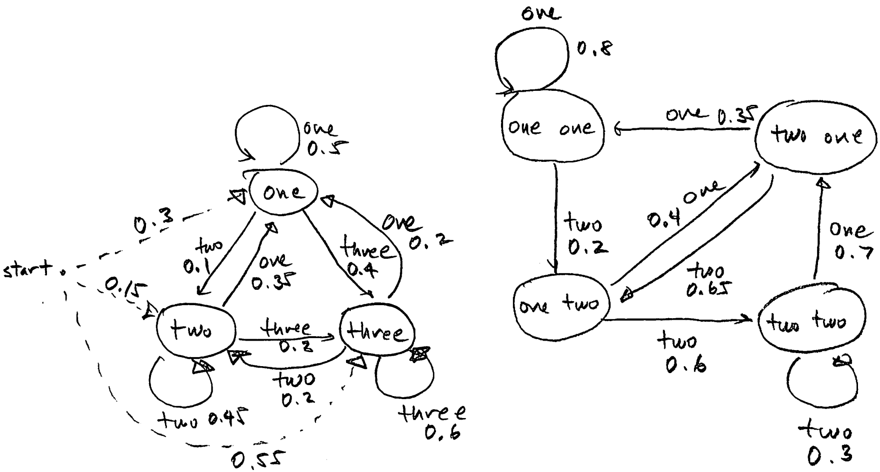


- Replace each word with phonetic word HMM.

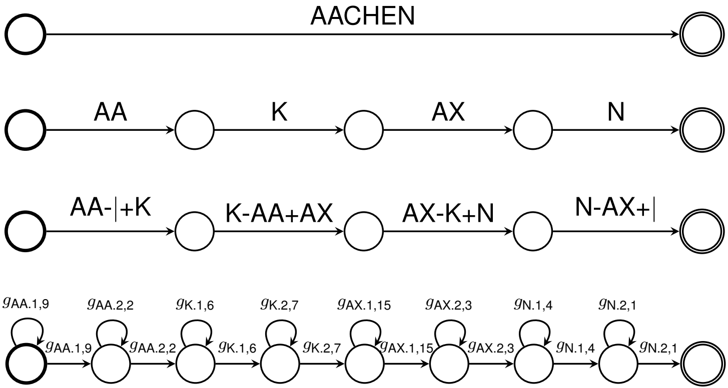




N-Gram Models as (H)MM's



Substituting in Word HMM's



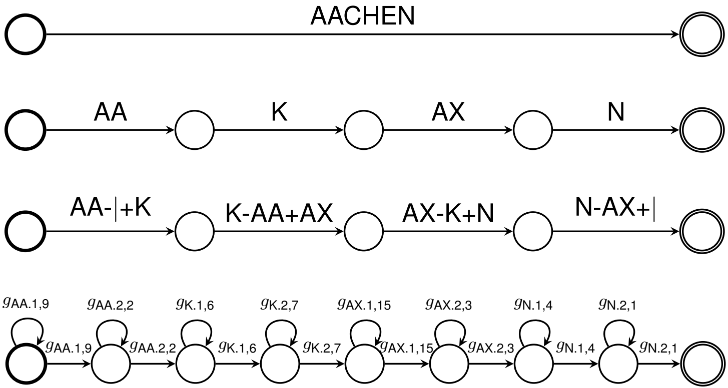
Recap: Small *vs.* Large Vocabulary Decoding

- It's all about building the one big HMM.
- Add in LM scores in graph; Viterbi unchanged.
- Start from word LM; substitute in word HMM's.

Where Are We?

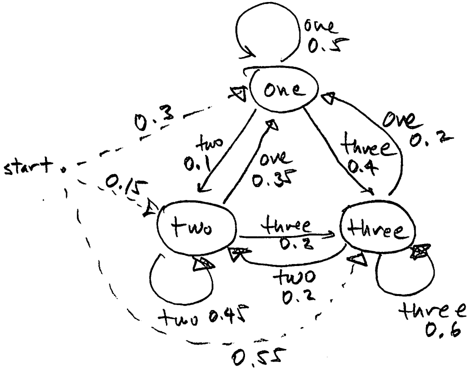
1. Introduction to FSA's, FST's, and Composition
2. What Can Composition Do?
3. How To Compute Composition
4. Composition and Graph Expansion
5. Weighted FSM's

Substituting in Word HMM's



- What about cross-word dependencies?
- *e.g.*, no boundary token; quinphones.

Cross-Word Dependencies



- Tricky: single-phone words; depend on *two* words away.

Graph Expansion Issues

- How to handle context-dependency?
- How to "glue in" HMM's, *e.g.*, word HMM's into an LM?
- How to do graph optimization?
- And handle scores/probs.
- Is there an elegant framework for all this?

Finite-State Machines!

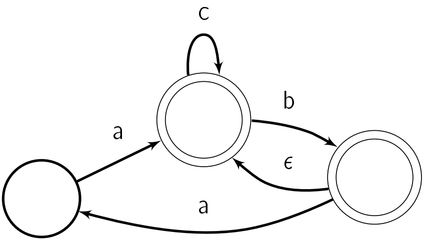
- A way of representing graphs/HMM's.
    - *e.g.*, LM's, one big HMM.
- A way of transforming graphs.
    - *e.g.*, substituting word HMM's into an LM.
- A set of graph *operations*.
    - *e.g.*, intersection, determinization, minimization, etc.
- *Weighted* graphs and transformations, too.

Graph Expansion and FSM's

- Design a bunch of "simple" finite-state machines.
- Apply standard FSM operations . . .
- To compute the one big HMM, and optimize it, too!

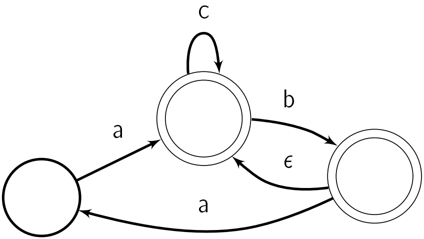
How To Represent a Graph/HMM?

- Finite-state *acceptor* (FSA).
- Just like HMM with symbolic outputs.
- Exactly one initial state; one or more final states.
- Arcs can be labeled with ϵ.
- Ignore probabilties for now.



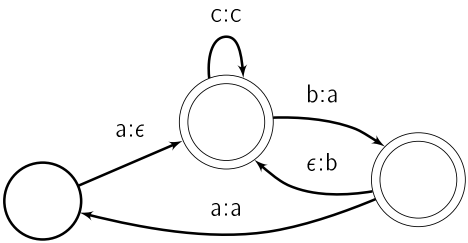
What Does an FSA Accept?

- An FSA *accepts* a string *i* . . .
- If path from initial to final state labeled with *i*.
- Does this FSA accept *abb*? *acccbaacc*? *aca*? ϵ?
- Can an FSA accept an infinite number of strings?



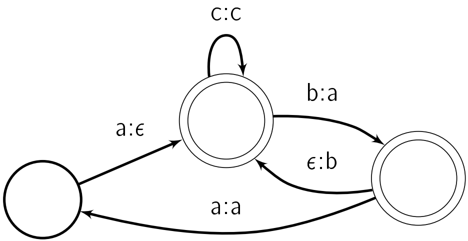
How To Represent a Graph *Transformation*?

- Finite-state *transducer* (FST).
- Like FSA, except each arc has *two* symbols.
  - An *input* label (possibly ϵ).
  - An *output* label (possibly ϵ).
- Intuition: rewrites input labels as output labels.



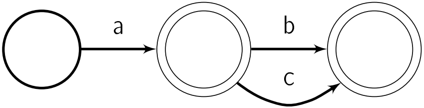
What Does an FST Accept?

- An FST accepts a string pair (i, o) …
- If path from initial to final state …
- Labeled with *i* on input side and *o* on output side.
- Does this FST accept (*acb*, *ca*)? (*acb*, *a*)?

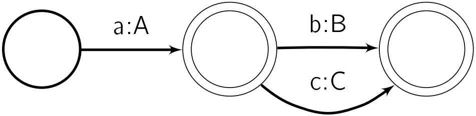


How To Apply a Graph Transformation?

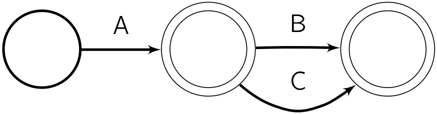
- *Composition*!
- Given FSA graph A, *e.g.*,



- And FST transformation T, *e.g.*,

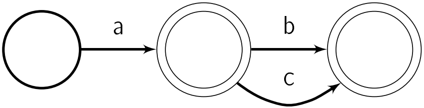


- Their composition $A \circ T$ is an FSA, *e.g.*,

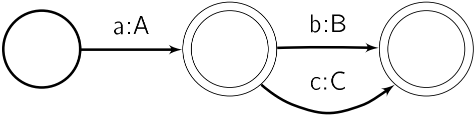


Composition Intuition

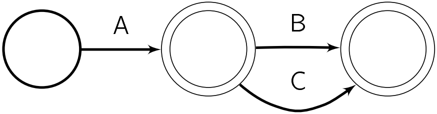
- If A accepts string i, *e.g.*, ab . . .



- And T accepts pair (i, o), *e.g.*, (ab, AB) . . .



- Then $A \circ T$ accepts string o, *e.g.*, AB.



- Perspective: trace paths in A and T together.

Recap

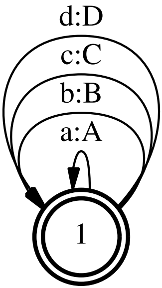
- Graphs: FSA's.
  - One label on each arc.
- Graph transformations: FST's.
  - Input *and* output label on each arc.
- Use *composition* to apply FST to FSA; produces FSA.

Where Are We?

1. Introduction to FSA's, FST's, and Composition
2. What Can Composition Do?
3. How To Compute Composition
4. Composition and Graph Expansion
5. Weighted FSM's

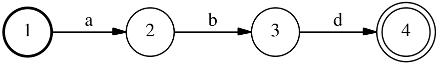
A Simple Class of FST's

- Replacing single symbol with single symbol, everywhere.

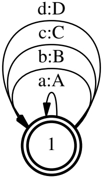


Rewriting Single String A Single Way

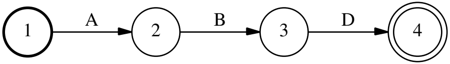
A



T

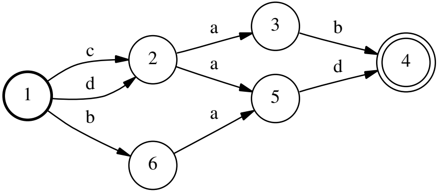


$A \circ T$

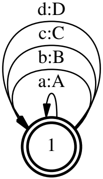


Rewriting Many Strings At Once

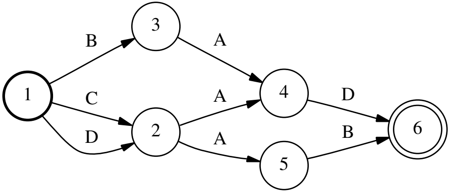
A



T

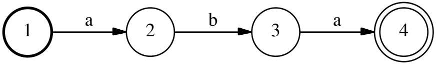


$A \circ T$

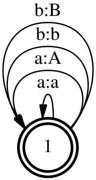


Rewriting Single String Many Ways

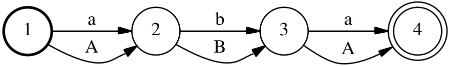
A



T

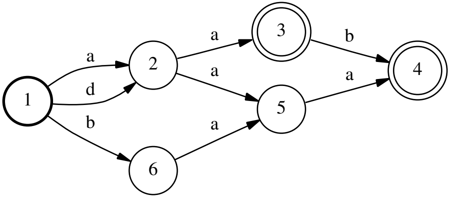


$A \circ T$



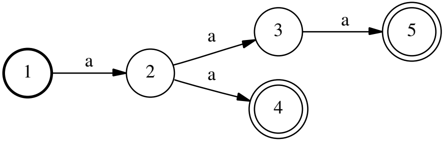
Rewriting Some Strings Zero Ways

A



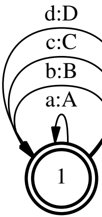
T

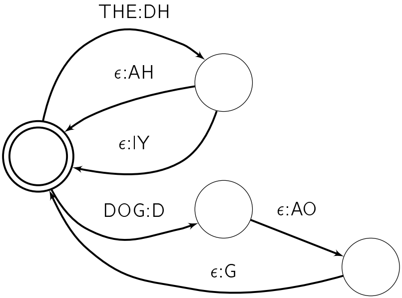
$A \circ T$



Generalizing Replacement

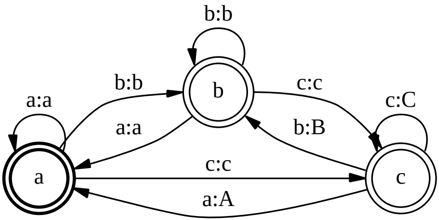
- Instead of replacing single symbol with single symbol . . .
- Can replace arbitrary string with arbitrary string.
- *e.g.*, what does FST on right do?





Context-Dependent Replacement

- Instead of *always* replacing symbol with symbol . . .
- Only do so in certain context.
- *e.g.*, what does this FST do? (Think: bigram model.)



Discussion

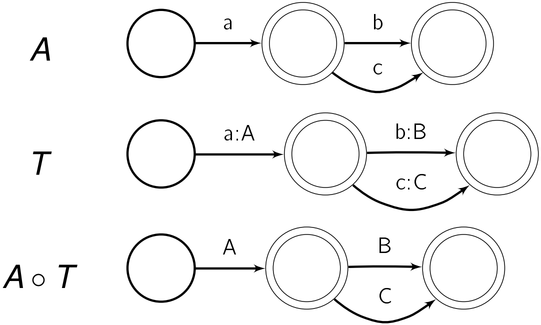
- Transforming a single string to a single string is *easy*.
  - *e.g.*, change *color* to *colour* everywhere in file.
- Composition: rewrites *every* string accepted by graph.
- Things composition can do:
  - Transform (possibly infinite) set of strings!
  - Not just 1:1, but 1:many and 1:0 transforms!
  - Can replace arbitrary strings with arbitrary strings!
  - Can do context-dependent transforms!
  - Expresses output compactly, as another graph!

Where Are We?

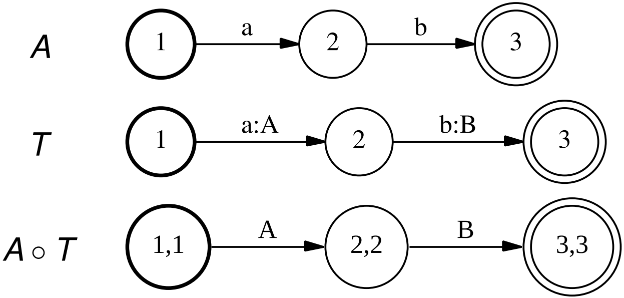
1. Introduction to FSA's, FST's, and Composition
2. What Can Composition Do?
3. How To Compute Composition
4. Composition and Graph Expansion
5. Weighted FSM's

How To Define Composition?

- $A \circ T$ accepts the string o iff ...
- There exists a string i such that ...
- A accepts i and T accepts (i, o).

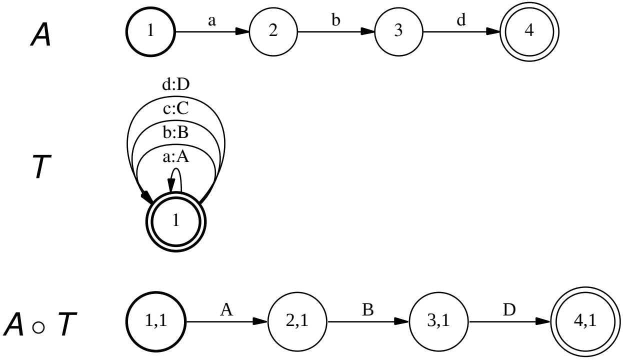


A Simple Case



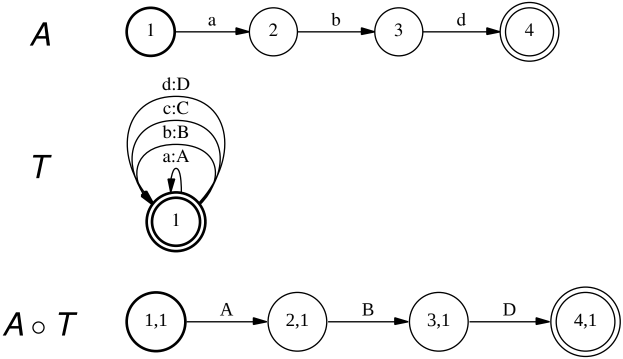
- Intuition: trace through A, T simultaneously.

Another Simple Case



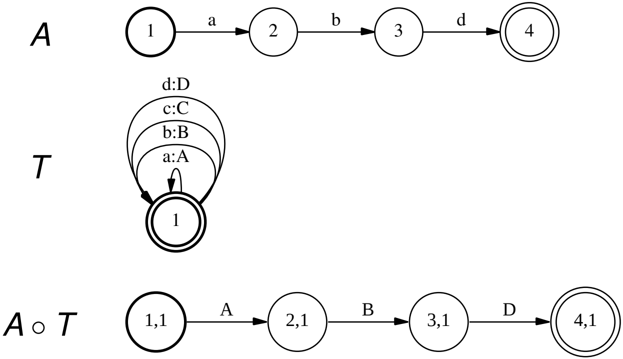
- Intuition: trace through A, T simultaneously.

Composition: States



- What is the possible set of states in result?
- Cross product of states in inputs, *i.e.*, (s_1, s_2).

Composition: Arcs

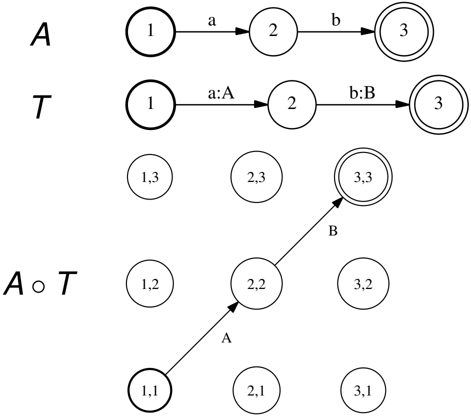


- Create arc from (s_1, t_1) to (s_2, t_2) with label o iff . . .
- Arc from s_1 to s_2 in A with label i and . . .
- Arc from t_1 to t_2 in T with input i and output o.

The Composition Algorithm

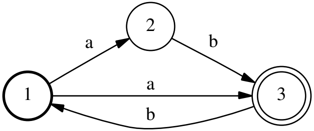
- For every state $s \in A$, $t \in T$, create state $(s, t) \in A \circ T$.
- Create arc from (s_1, t_1) to (s_2, t_2) with label o iff ...
  - Arc from s_1 to s_2 in A with label i and ...
  - Arc from t_1 to t_2 in T with input i and output o.
- (s, t) is initial iff s and t are initial; similarly for final states.
- What is time complexity?

Example

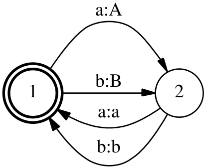


Another Example

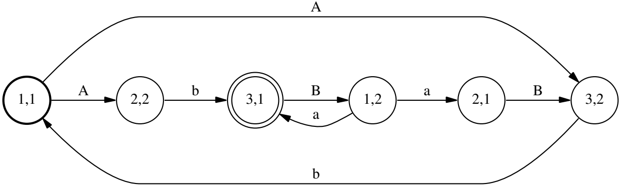
A



T



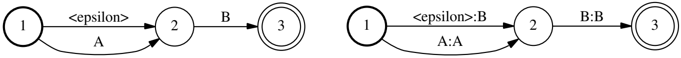
$A \circ T$



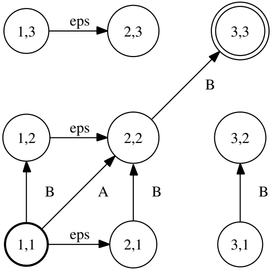
Composition and ϵ-Transitions

- Basic idea: can take ϵ-transition in one FSM ...
  - Without moving in other FSM.
- Tricky to do exactly right.
  - Do readings if you care: (Pereira, Riley, 1997)

A, T



$A \circ T$



Recap

- Composition is easy!
- Composition is fast!
- Worst case: quadratic in states.
  - Optimization: only expand reachable state pairs.

Where Are We?

1 Introduction to FSA's, FST's, and Composition

2 What Can Composition Do?

3 How To Compute Composition

4 Composition and Graph Expansion

5 Weighted FSM's

Building the One Big HMM

- Can we do this with composition?
- Start with *n*-gram LM expressed as HMM.
- Repeatedly expand to lower-level HMM's.

A View of Graph Expansion

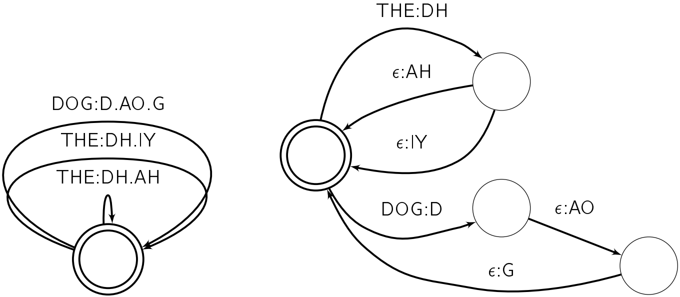
- Design some finite-state machines.
  - L = language model FSA.
  - $T_{\text{LM}\to\text{CI}}$ = FST mapping to CI phone sequences.
  - $T_{\text{CI}\to\text{CD}}$ = FST mapping to CD phone sequences.
  - $T_{\text{CD}\to\text{GMM}}$ = FST mapping to GMM sequences.
- Compute final decoding graph via composition:

$$L \circ T_{\text{LM}\to\text{CI}} \circ T_{\text{CI}\to\text{CD}} \circ T_{\text{CD}\to\text{GMM}}$$

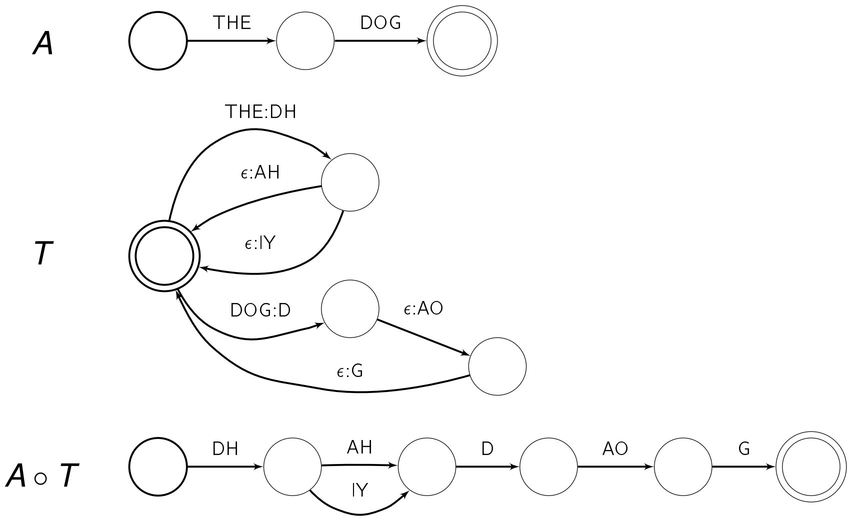
- How to design transducers?

Example: Mapping Words To Phones

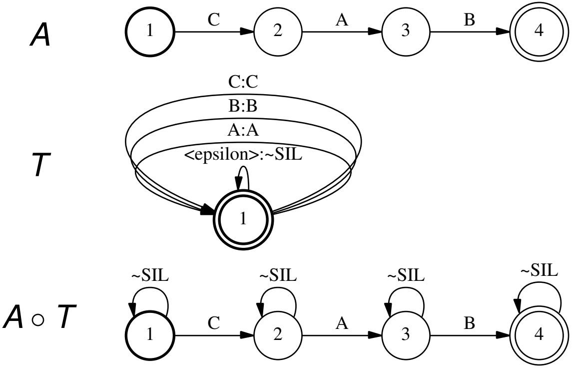
THE	DH AH
THE	DH IY
DOG	D AO G



Example: Mapping Words To Phones



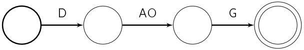
Example: Inserting Optional Silences



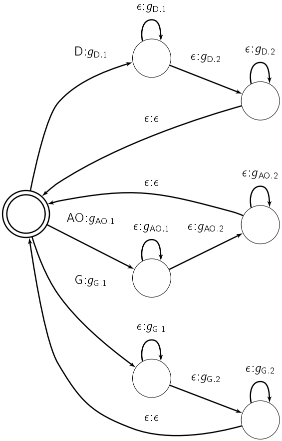
- Don't forget identity transformations!
- Strings that aren't accepted are discarded.

Example: Rewriting CI Phones as HMM's

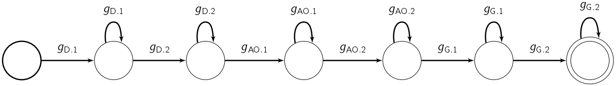
A



T

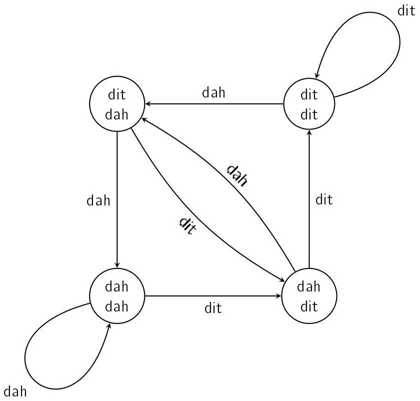


$A \circ T$

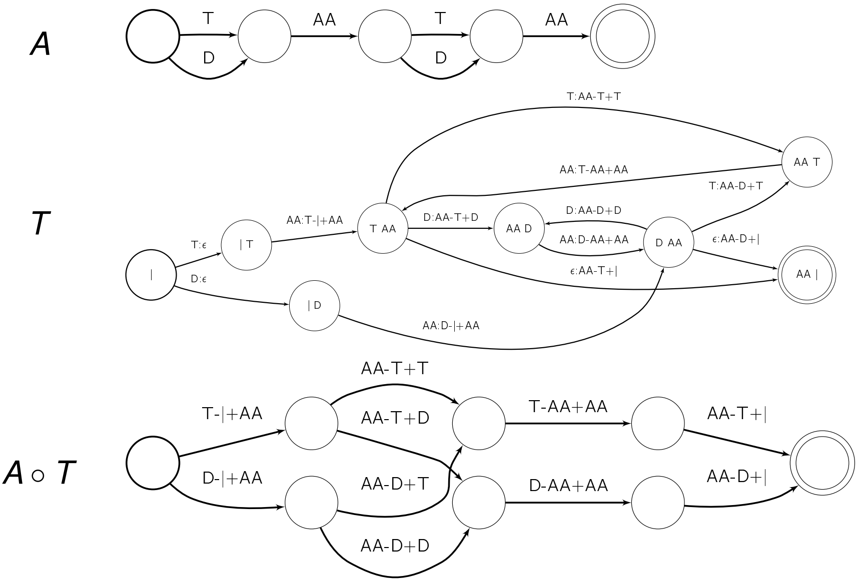


Example: Rewriting CI $\Rightarrow$ CD Phones

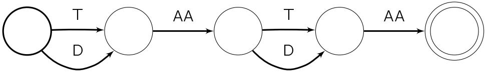
- *e.g.*, $\texttt{L} \Rightarrow \texttt{L-S+IH}$
- The basic idea: adapt FSA for trigram model.
- When take arc, know current trigram ($P(w_i|w_{i-2}w_{i-1})$).
- Output $w_{i-1}-w_{i-2}+w_i$!

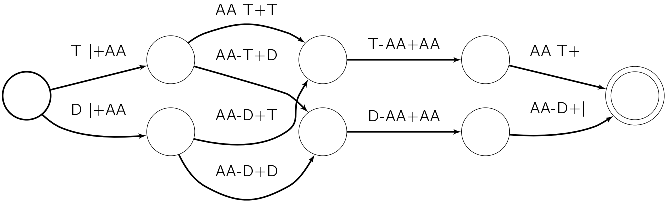


How to Express CD Expansion via FST's?



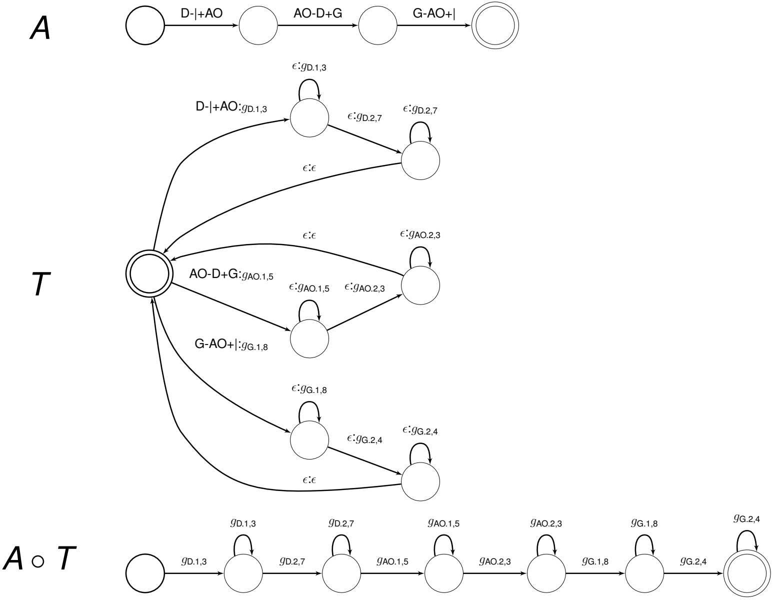
How to Express CD Expansion via FST's?





- Point: composition automatically expands FSA . . .
  - To correctly handle context!
- Makes multiple copies of states in original FSA . . .
  - That can exist in different triphone contexts.
  - (And makes multiple copies of *only* these states.)

Example: Rewriting CD Phones as HMM's



Recap: Whew!

- Design some finite-state machines.
  - L = language model FSA.
  - $T_{\mathrm{LM}\rightarrow\mathrm{CI}}$ = FST mapping to CI phone sequences.
  - $T_{\mathrm{CI}\rightarrow\mathrm{CD}}$ = FST mapping to CD phone sequences.
  - $T_{\mathrm{CD}\rightarrow\mathrm{GMM}}$ = FST mapping to GMM sequences.
- Compute final decoding graph via composition:

$$L \circ T_{\mathrm{LM}\rightarrow\mathrm{CI}} \circ T_{\mathrm{CI}\rightarrow\mathrm{CD}} \circ T_{\mathrm{CD}\rightarrow\mathrm{GMM}}$$

Where Are We?

1 Introduction to FSA's, FST's, and Composition

2 What Can Composition Do?

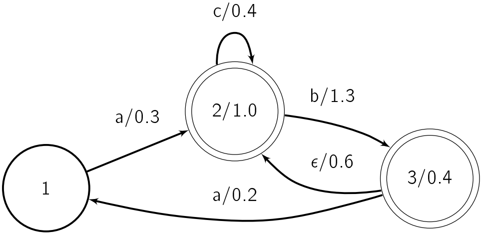
3 How To Compute Composition

4 Composition and Graph Expansion

5 Weighted FSM's

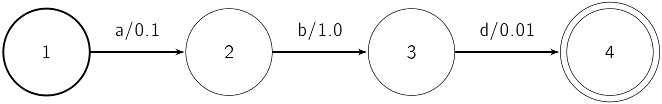
What About Those Probability Thingies?

- *e.g.*, to hold language model probs, transition probs, etc.
- FSM's ⇒ *weighted* FSM's.
  - WFSA's, WFST's.
- Each arc has score or *cost*.
  - So do final states.

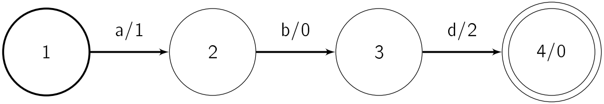


What Is A Cost?

- HMM's have probabilities on arcs.
  - Prob of path is product of arc probs.

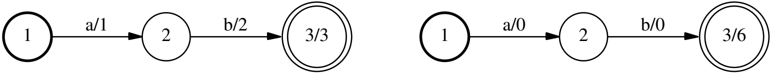


- WFSM's have negative log probs on arcs.
  - Cost of path is sum of arc costs plus final cost.



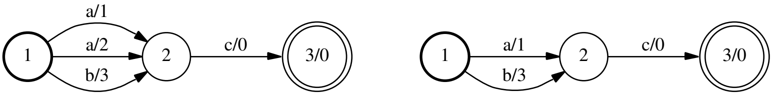
What Does a WFSA Accept?

- A WFSA *accepts* a string i with cost c …
- If path from initial to final state labeled with i and with cost c.
- How costs/labels distributed along path doesn't matter!
- Do these accept same strings with same costs?



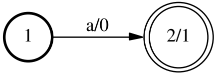
What If Two Paths With Same String?

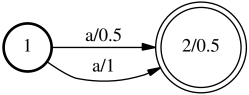
- How to compute cost for this string?
- Use "min" operator to compute combined cost?
    - Combine paths with same labels.

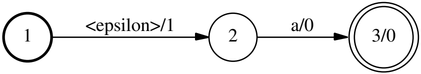


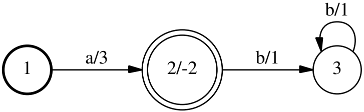
- Operations $(+, \min)$ form a *semiring* (the *tropical* semiring).

Which Is Different From the Others?

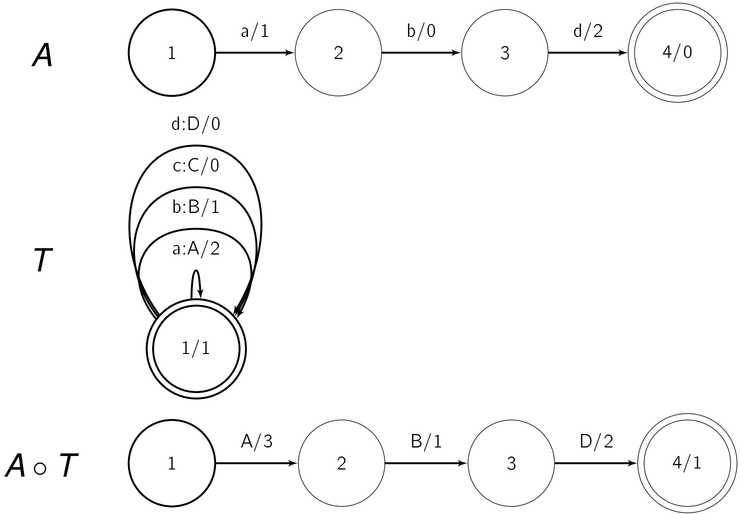








Weighted Composition



The Bottom Line

- Place LM, AM log probs in L, $T_{\text{LM}\to\text{CI}}$, $T_{\text{CI}\to\text{CD}}$, $T_{\text{CD}\to\text{GMM}}$.
  - *e.g.*, LM probs, pronunciation probs, transition probs.
- Compute decoding graph via weighted composition:

$$L \circ T_{\text{LM}\to\text{CI}} \circ T_{\text{CI}\to\text{CD}} \circ T_{\text{CD}\to\text{GMM}}$$

- Then, doing Viterbi decoding on this big HMM ...
  - Correctly computes (more or less):

$$\omega^* = \arg\max_{\omega} P(\omega|\mathbf{x}) = \arg\max_{\omega} P(\omega)P(\mathbf{x}|\omega)$$

Recap: FST's and Composition? Awesome!

- Operates on all paths in WFSA (or WFST) simultaneously.
- Rewrites symbols as other symbols.
- Context-dependent rewriting of symbols.
- Adds in new scores.
- Restricts set of allowed paths (intersection).
- Or all of above at once.

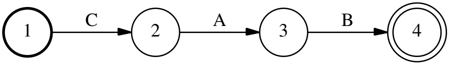
Weighted FSM's and ASR

- Graph expansion can be framed . . .
  - As series of (weighted) composition operations.
- Correctly combines scores from multiple WFSM's.
- Building FST's for each step is pretty straightforward . . .
  - Except for context-dependent phone expansion.
- Handles graph expansion for training, too.

Discussion

- Don't need to write code?!
  - AT&T FSM toolkit ⇒ OpenFST; lots of others.
  - Generate FST's as text files.

```
1    2    C
2    3    A
3    4    B
4
```



- WFSM framework is very flexible.
  - Just design new FST's!
  - *e.g.*, CD pronunciations at word or phone level.

Part II

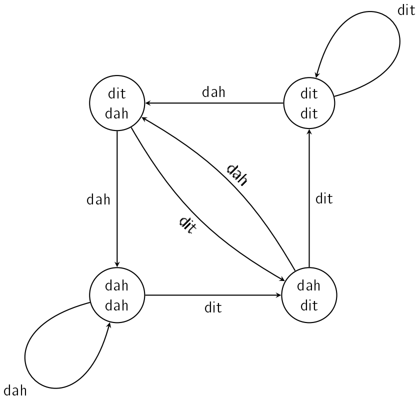
Making Decoding Fast

How Big? How Fast?

- Time to look at efficiency.
- How big is the one big HMM?
- How long will Viterbi take?

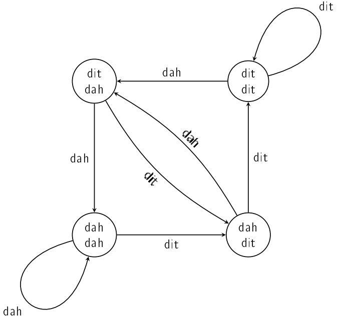
Pop Quiz

- How many states in HMM representing trigram model . . .
  - With vocabulary size $|V|$?
- How many arcs?



Issue: How Big The Graph?

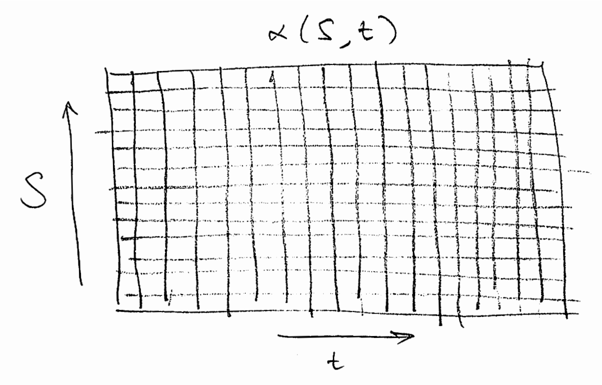
- Trigram model (*e.g.*, vocabulary size $|V| = 2$)



- $|V|^3$ word arcs in FSA representation.
- Words are $\sim$4 phones = 12 states on average (CI).
- If $|V| = 50000$, $50000^3 \times 12 \approx 10^{15}$ states in graph.
- PC's have $\sim 10^{10}$ bytes of memory.

Issue: How Slow Decoding?

- In each frame, loop through every state in graph.
- If 100 frames/sec, 10^{15} states ...
  - How many cells to compute per second?
- A core can do $\sim 10^{11}$ floating-point ops per second.



Recap

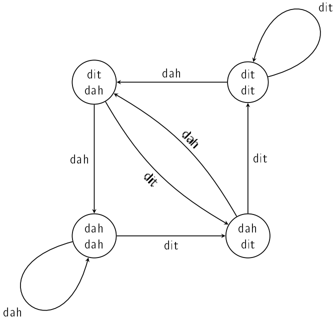
- Naive graph expansion is way too big; Viterbi way too slow.
- Shrinking the graph also makes things faster!
- How to shrink the one big HMM?

Where Are We?

1. Shrinking the Language Model
2. Graph Optimization
3. Pruning
4. Other Viterbi Optimizations
5. Other Decoding Paradigms

Compactly Representing N-Gram Models

- One big HMM size $\propto$ LM HMM size.
- Trigram model: $|V|^3$ arcs in naive representation.



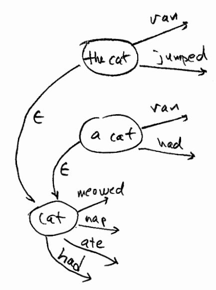
- Small fraction of all trigrams occur in training data.
  - Is it possible to keep arcs only for seen trigrams?

Compactly Representing *N*-Gram Models

- Can express smoothed *n*-gram models ...
  - Via backoff distributions.

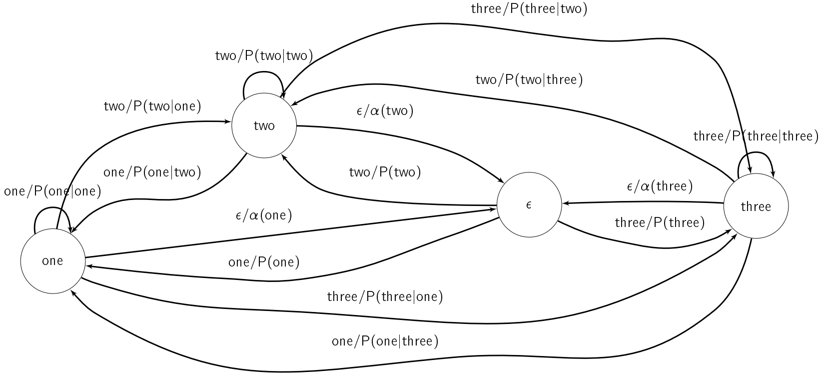
$$P_{\text{smooth}}(w_i|w_{i-1}) = \begin{cases} P_{\text{primary}}(w_i|w_{i-1}) & \text{if } \operatorname{count}(w_{i-1}w_i) > 0 \\ \alpha_{w_{i-1}} P_{\text{smooth}}(w_i) & \text{otherwise} \end{cases}$$

- Idea: avoid arcs for unseen trigrams via *backoff* states.



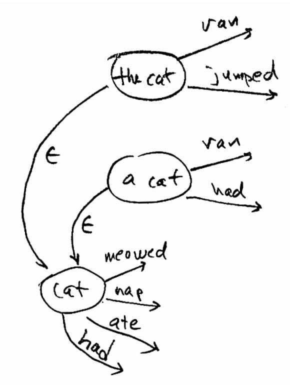
Compactly Representing *N*-Gram Models

$$P_{\text{smooth}}(w_i|w_{i-1}) = \begin{cases} P_{\text{primary}}(w_i|w_{i-1}) & \text{if } \text{count}(w_{i-1}w_i) > 0 \\ \alpha_{w_{i-1}} P_{\text{smooth}}(w_i) & \text{otherwise} \end{cases}$$



Problem Solved!?

- Is this FSA deterministic?
  - *i.e.*, are there multiple paths with same label sequence?
- Is this method *exact*?
  - Does Viterbi ever use the wrong probability?



Can We Make the LM Even Smaller?

- Sure, just remove some more arcs. Which?
- Count cutoffs.
  - *e.g.*, remove all arcs corresponding to n-grams . . .
  - Occurring fewer than k times in training data.
- Likelihood/entropy-based pruning (Stolcke, 1998).
  - Choose those arcs which when removed, . . .
  - Change likelihood of training data the least.

Discussion

- Only need to keep seen *n*-grams in LM graph.
  - Exact representation blows up graph several times.
- Can further prune LM to arbitrary size.
  - *e.g.*, for BN 4-gram model, 100MW training data ...
  - Pruning by factor of 50 $\Rightarrow$ +1% absolute WER.
- Graph small enough now?
  - Let's keep on going; smaller $\Rightarrow$ faster!

Where Are We?

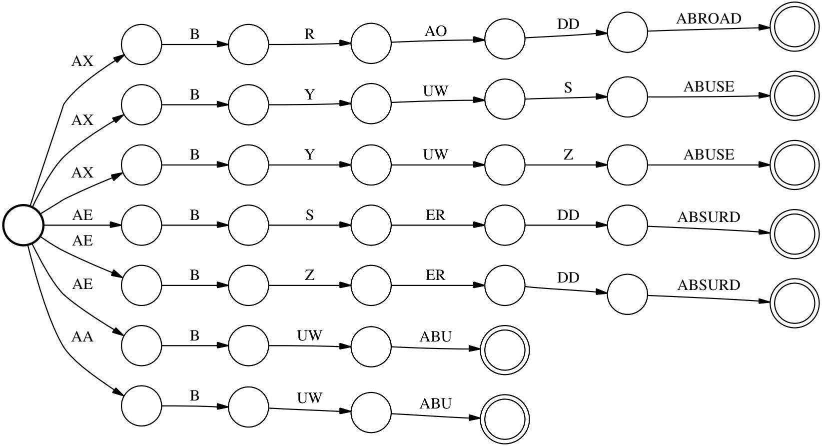
1. Shrinking the Language Model
2. Graph Optimization
3. Pruning
4. Other Viterbi Optimizations
5. Other Decoding Paradigms

Graph Optimization

- Can we modify topology of graph ...
- Such that it's smaller (fewer arcs or states) ...
- Yet accepts same strings (with same costs)?
- (OK to move labels and costs along paths.)

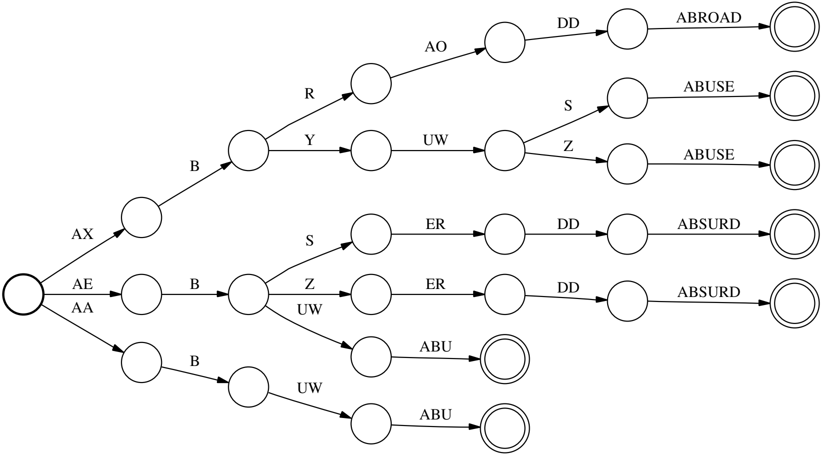
Graph Compaction

- Consider word graph for isolated word recognition.
  - Expanded to phone level: 39 states, 38 arcs.



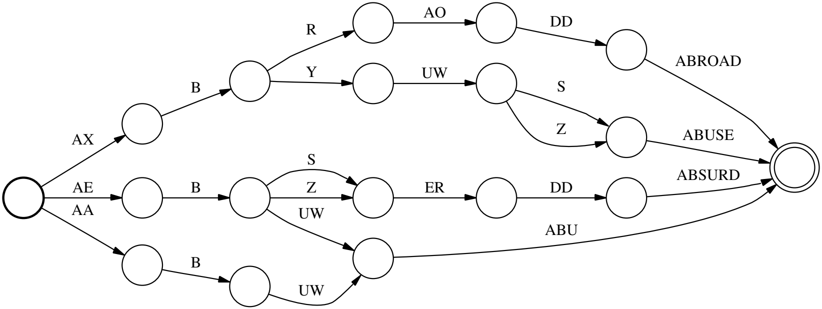
Determinization

- Share common prefixes: 29 states, 28 arcs.



Minimization

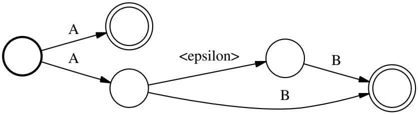
- Share common suffixes: 18 states, 23 arcs.

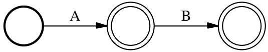


- Does this accept same strings as original graph?
- Original: 39 states, 38 arcs.

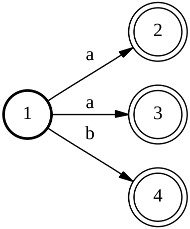
What Is A Deterministic FSM?

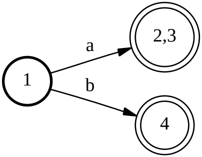
- Same as being *nonhidden* for HMM.
- No two arcs exiting same state with same input label.
- No ϵ arcs.
- *i.e.*, for any input label sequence . . .
  - Only one state reachable from start state.





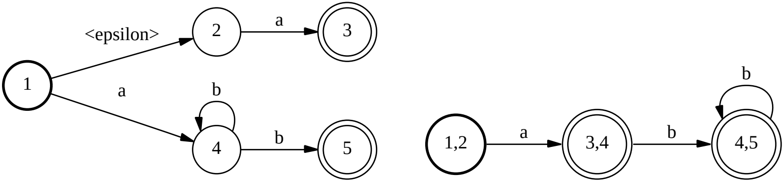
Determinization: A Simple Case





- Does this accept same strings?
- States on right $\Leftrightarrow$ state *sets* on left!

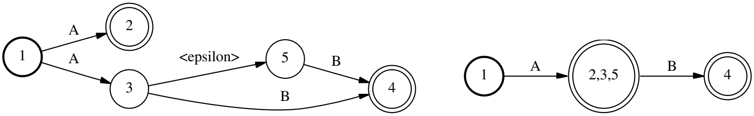
A Less Simple Case



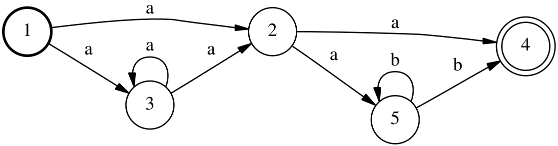
- Does this accept same strings? (ab^*)

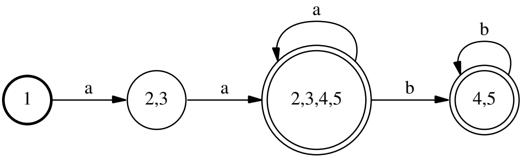
Determinization

- Start from start state.
- Keep list of state sets not yet expanded.
  - For each, compute outgoing arcs in logical way . . .
  - Creating new state sets as needed.
- Must follow ϵ arcs when computing state sets.

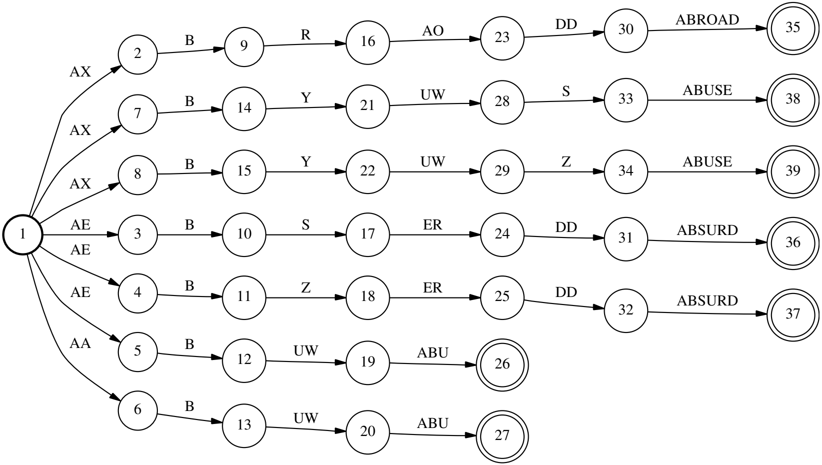


Example 2



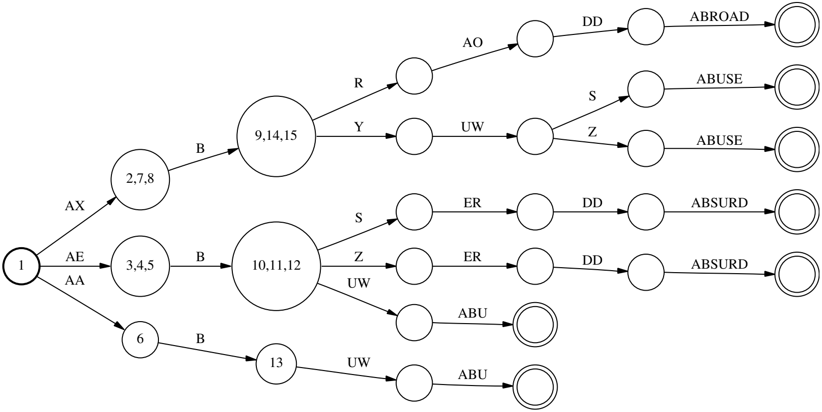


Example 3



1
AX
AX
AX
AE
AE
AE
AA
2
7
8
3
4
5
6
B
9
14
15
10
11
12
13
R
Y
Y
S
Z
UW
UW
16
21
22
17
18
19
20
AO
UW
UW
ER
ER
ABU
ABU
23
28
29
24
25
26
27
DD
S
Z
DD
DD
30
33
34
31
32
ABROAD
ABUSE
ABUSE
ABSURD
ABSURD
35
38
39
36
37

Example 3, Continued



Pop Quiz: Determinization

- For FSA with s states, ...
  - What is max number of states when determinized?
  - *i.e.*, how many possible unique state sets?
- Are all unweighted FSA's determinizable?
  - *i.e.*, does algorithm always terminate ...
  - To produce equivalent deterministic FSA?

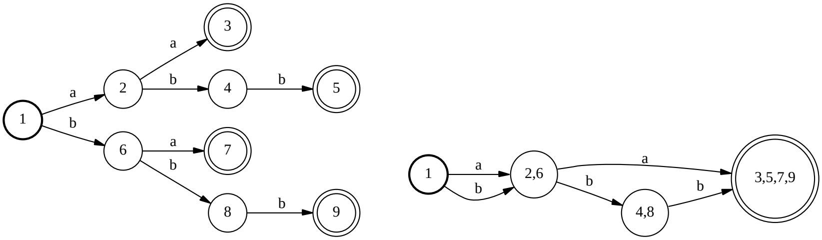
Minimization

- What should we minimize?
- The number of states!

Minimization Basics

- Algorithm only correct for deterministic FSM's.
- Output FSM is also deterministic.
- Basic idea: suffix sharing.
  - Can merge two states if have same "suffix".

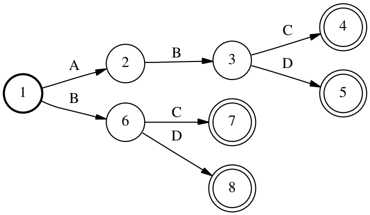
Minimization: A Simple Case

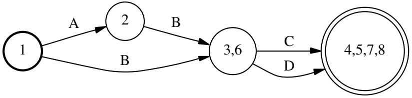


- Does this accept same strings?
- States on right $\Leftrightarrow$ state *sets* on left! Partition!

Minimization: Acyclic Graphs

- Merge states with same following strings (*follow sets*).

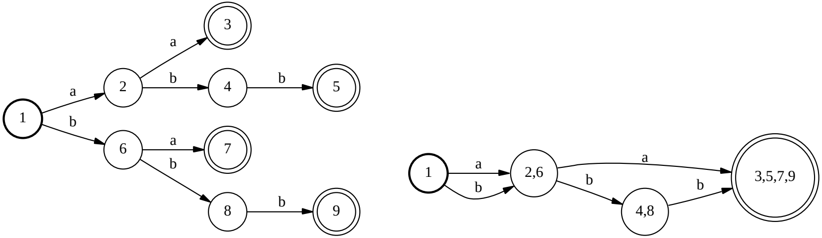




states	following strings
1	ABC, ABD, BC, BD
2	BC, BD
3, 6	C, D
4,5,7,8	ϵ

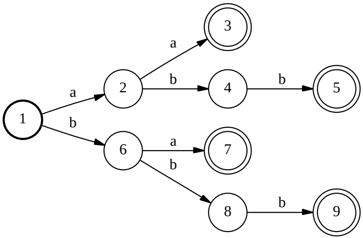
General Minimization: The Basic Idea

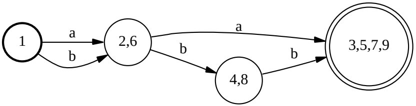
- Given deterministic FSM . . .
- Start with all states in single partition.
- Whenever states within partition . . .
  - Have “different” outgoing arcs or finality . . .
  - Split partition.
- At end, each partition corresponds to state in output FSM.
  - Make arcs in logical manner.



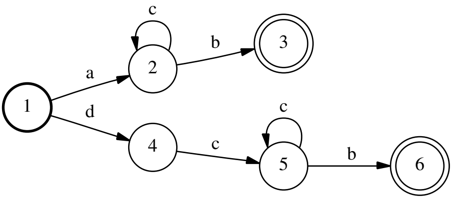
Minimization

- Invariant: if two states are in different partitions . . .
  - They have different follow sets.
- First split: final and non-final states.
  - Final states have ϵ in their follow sets.
- Two states in same partition have different follow sets if . . .
  - Different number of outgoing arcs or arc labels . . .
  - Or arcs go to different partitions.

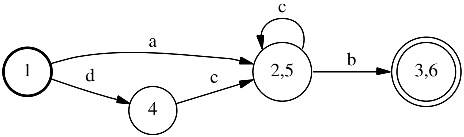




Minimization



action	evidence	partitioning
		{1,2,3,4,5,6}
split 3,6	final	{1,2,4,5}, {3,6}
split 1	has *a* arc	{1}, {2,4,5}, {3,6}
split 4	no *b* arc	{1}, {4}, {2,5}, {3,6}



Discussion

- Determinization.
  - May reduce or increase number of states.
  - Improves behavior of search $\Rightarrow$ prefix sharing!
- Minimization.
  - Minimizes states, not arcs, for deterministic FSM's.
  - Does minimization always terminate? How long?
- *Weighted* algorithms exist for both FSA's, FST's.
  - Available in FSM toolkits.
- Weighted minimization requires *push* operation.
  - Normalizes locations of costs/labels along paths . . .
  - So arcs that can be merged have same cost/label.

Weighted Graph Expansion, Optimized

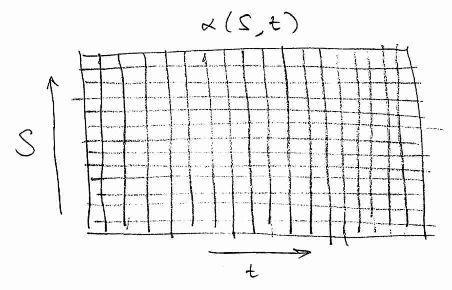
- Final graph: min(det($L \circ T_{\text{LM}\rightarrow\text{CI}} \circ T_{\text{CI}\rightarrow\text{CD}} \circ T_{\text{CD}\rightarrow\text{GMM}}$))
  - L = pruned, backoff language model FSA.
  - $T_{\text{LM}\rightarrow\text{CI}}$ = FST mapping to CI phone sequences.
  - $T_{\text{CI}\rightarrow\text{CD}}$ = FST mapping to CD phone sequences.
  - $T_{\text{CD}\rightarrow\text{GMM}}$ = FST mapping to GMM sequences.
- Build big graph; minimize at end?
  - Problem: can't hold big graph in memory.
  - Many existing recipes for graph expansion.
- $10^{15}+$ states $\Rightarrow$ 20–50M states/arcs.
  - 5–10M *n*-grams kept in LM.

Where Are We?

1. Shrinking the Language Model
2. Graph Optimization
3. Pruning
4. Other Viterbi Optimizations
5. Other Decoding Paradigms

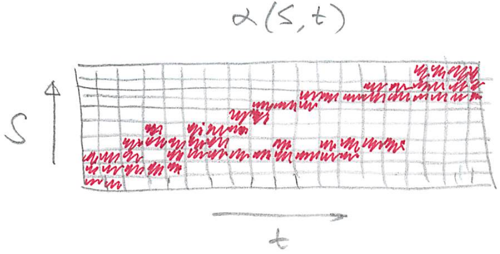
Real-Time Decoding

- Why is this desirable?
- Decoding time for Viterbi algorithm; 10M states in graph.
  - 100 frames/sec $\times$ 10M states $\times$ ...
  - 100 cycles/state $\Rightarrow 10^{11}$ cycles/sec.
  - PC's do $\sim 10^9$ cycles/second (*e.g.*, 3GHz Xeon).
- Cannot afford to evaluate each state at each frame.
  - Need to optimize Viterbi algorithm!



Pruning

- At each frame, only evaluate cells with highest scores.
- Given *active* states/cells from last frame ...
  - Only examine states/cells in current frame ...
  - Reachable from active states in last frame.
  - Keep best to get active states in current frame.



Don’t Throw Out the Baby

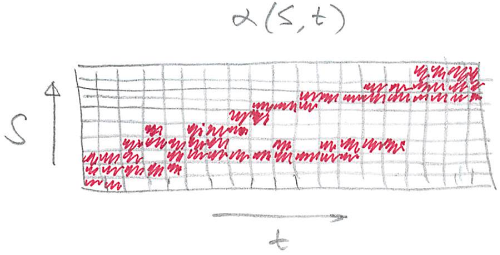
- When not considering every state at each frame ...
  - Can make *search errors*.

$$\omega^* = \arg\max_{\omega} P(\omega|\mathbf{x}) = \arg\max_{\omega} P(\omega)P(\mathbf{x}|\omega)$$

- The goal of *search*:
  - Minimize computation *and* search errors.

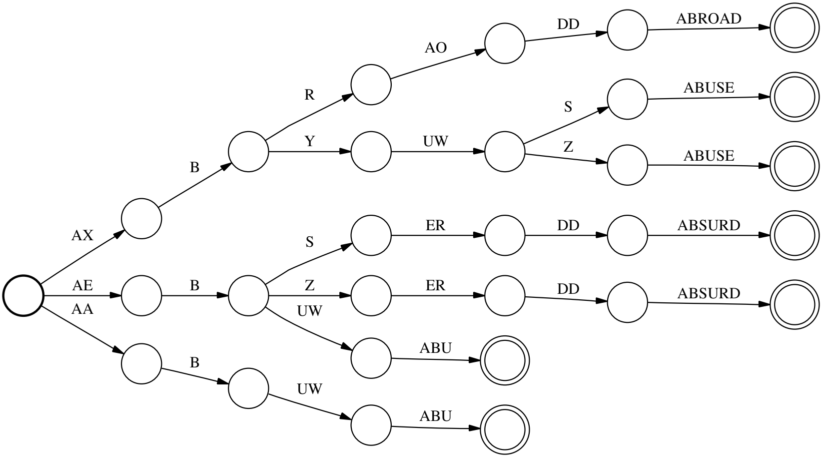
How Many Active States To Keep?

- Goal: Prune paths with no chance of becoming *best* path.
- *Beam* pruning.
  - Keep only states with log probs within fixed distance . . .
  - Of best log prob at that frame.
- *Rank* or *histogram* pruning.
  - Keep only *k* highest scoring states.
- When are these good? Bad? Can get best of both?



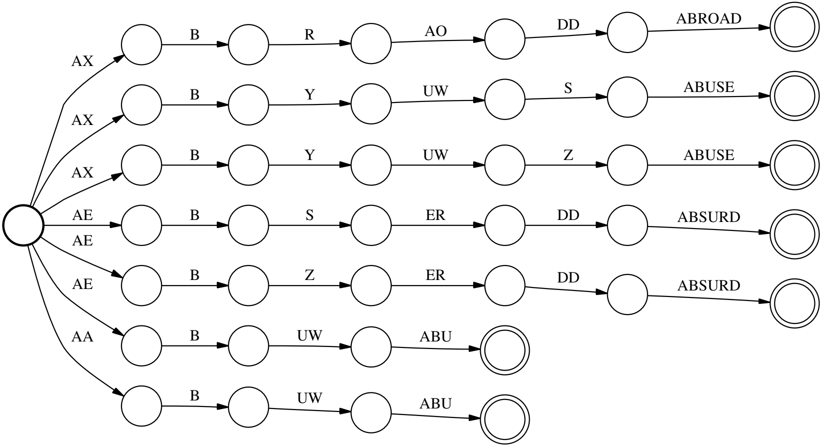
Pruning Visualized

- Active states are small fraction of total states (<1%)
- Tend to be localized in small regions in graph.



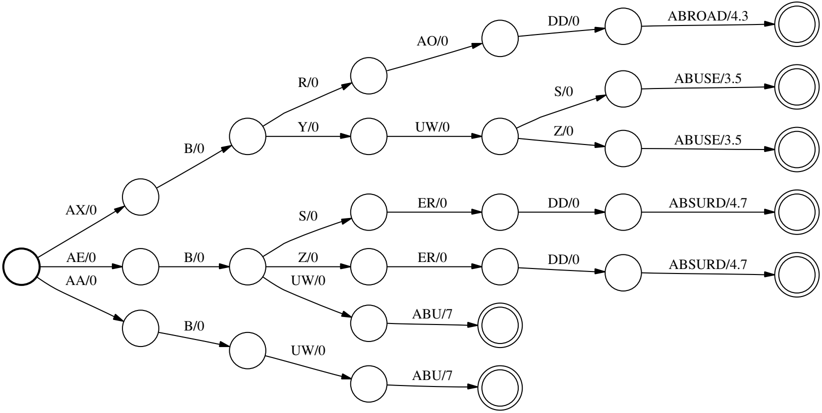
Pruning and Determinization

- Most uncertainty occurs at word starts.
- Determinization drastically reduces branching here.



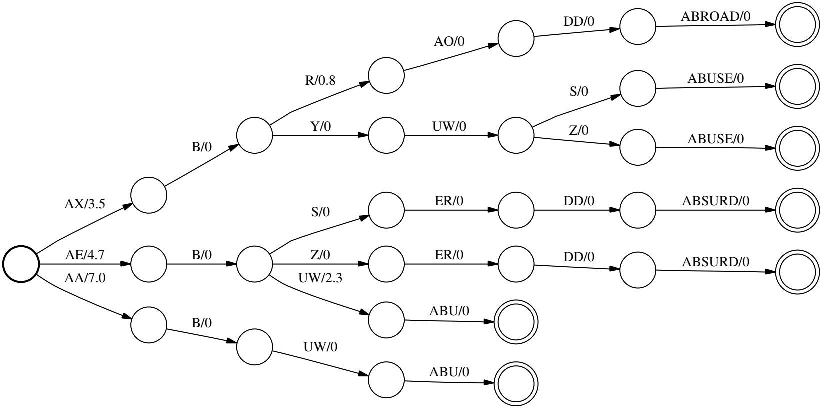
Language Model Lookahead

- In practice, put word labels at word ends. (Why?)
- What's wrong with this picture? (Hint: think beam pruning.)



Language Model Lookahead

- Move LM scores as far ahead as possible.
- At each point, total cost $\Leftrightarrow$ min LM cost of following words.
- *push* operation does this.

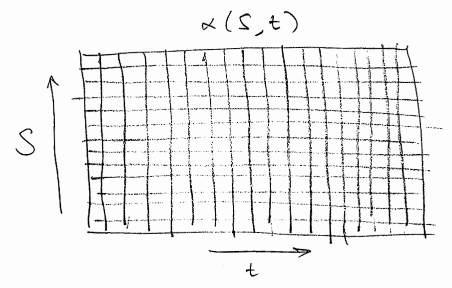


Where Are We?

1. Shrinking the Language Model
2. Graph Optimization
3. Pruning
4. Other Viterbi Optimizations
5. Other Decoding Paradigms

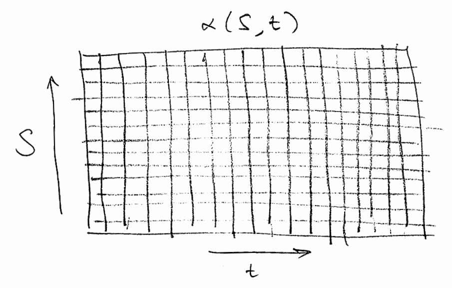
Saving Memory

- Naive Viterbi implementation: store whole DP chart.
- If 10M-state decoding graph:
  - 10 second utterance $\Rightarrow$ 1000 frames.
  - 1000 frames $\times$ 10M states = 10 billion cells.
- Each cell holds:
  - Viterbi log prob; backtrace pointer.

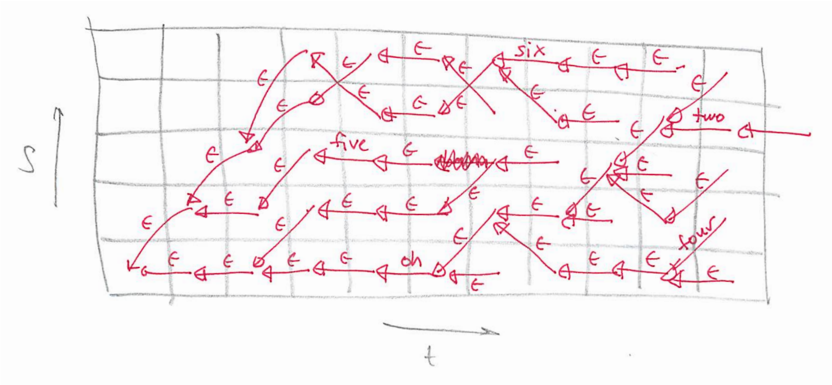


Forgetting the Past

- To compute cells at frame t ...
  - Only need cells at frame $t - 1$!
- Only reason need to keep cells from past ...
  - Is for backtracing, to recover word sequence.
- Can we store backtracing information another way?

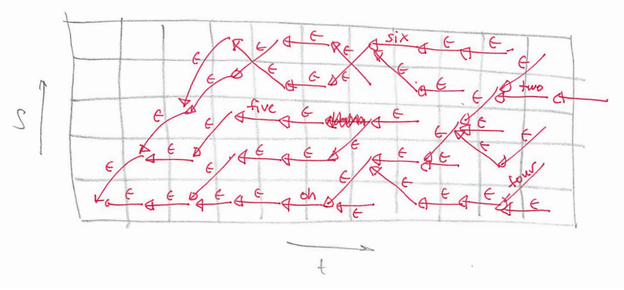


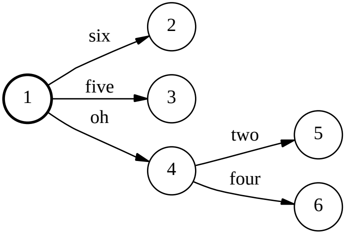
Compressing Backtraces



- Only need to remember *graph*! (Can forget gray stuff.)
- How to make this graph smaller?

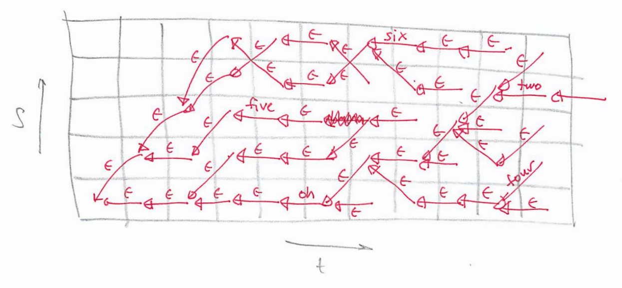
Determinization!

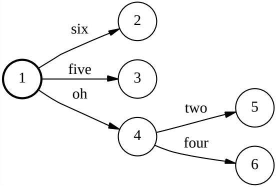


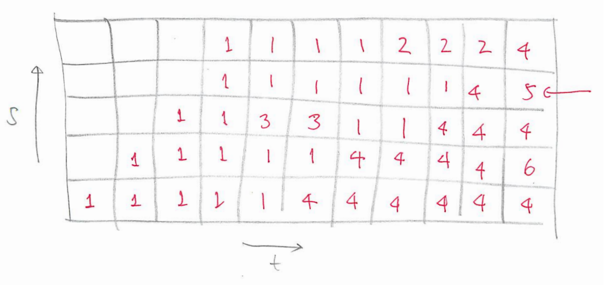


- In each cell, just remember node in FSA!

Token Passing

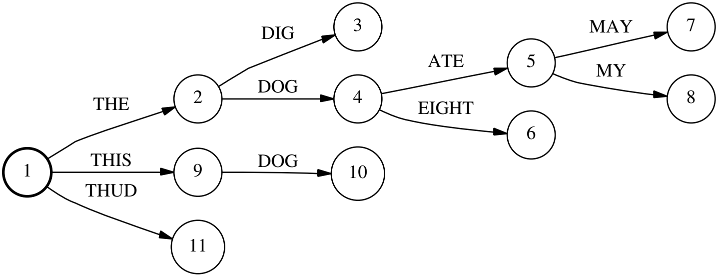






Token Passing

- Maintain "word tree":
  - Node represents word sequence from start state.
- Backtrace pointer points to node in tree ...
  - Holding word sequence labeling best path to cell.
- Set backtrace to same node as at best last state ...
  - Unless cross word boundary.



Recap: Efficient Viterbi Decoding

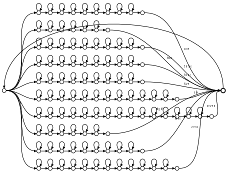
- The essence: one big HMM and Viterbi.
- Graph optimization crucial, but not enough by itself.
- Pruning is key for speed.
  - Determinization and LM lookahead help pruning a ton.
- Can process $\sim$10000 states/frame in $<1\times$ RT on PC.
  - Can process $\sim$1% of cells for 10M-state graph …
  - And make very few search errors.
- Depending on application and resources …
  - May run faster or slower than $1\times$ RT (desktop).
- Memory usage.
  - The biggie: decoding graph (shared memory).

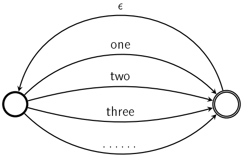
Where Are We?

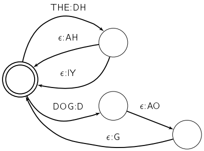
1. Shrinking the Language Model
2. Graph Optimization
3. Pruning
4. Other Viterbi Optimizations
5. Other Decoding Paradigms

My Language Model Is Too Small

- What we've described: *static* graph expansion.
  - To make decoding graph tractable . . .
  - Use heavily-pruned language model.
- Another approach: *dynamic* graph expansion.
  - Don't store whole graph in memory.
  - Build parts of graph with active states on the fly.







Dynamic Graph Expansion: The Basic Idea

- Express graph as composition of two smaller graphs.
  - Composition is associative.

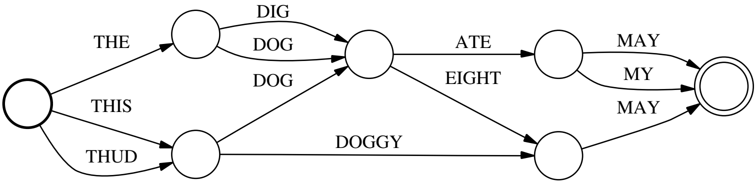
$$
\begin{aligned}
G_{\text{decode}} &= L \circ T_{\text{LM}\to\text{CI}} \circ T_{\text{CI}\to\text{CD}} \circ T_{\text{CD}\to\text{GMM}} \\
&= L \circ (T_{\text{LM}\to\text{CI}} \circ T_{\text{CI}\to\text{CD}} \circ T_{\text{CD}\to\text{GMM}})
\end{aligned}
$$

- Can do *on-the-fly* composition.
  - States in result correspond to state pairs (s_1, s_2).

Two-Pass Decoding

- What about my fuzzy logic 15-phone acoustic model . . .
  - And 7-gram neural net LM with SVM boosting?
- Some of the models developed in research are . . .
  - Too expensive to implement in one-pass decoding.
- First-pass decoding: use simpler model . . .
  - To find “likeliest” word *sequences* . . .
  - As lattice (WFSA) or flat list of hypotheses (*N*-best list).
- *Rescoring*: use complex model . . .
  - To find best word sequence . . .
  - Among first-pass hypotheses.

Lattice Generation and Rescoring



- In Viterbi, store *k*-best tracebacks at each word-end cell.
- To add in new LM scores to lattice . . .
  - What operation can we use?
- Lattices have other uses.
  - *e.g.*, confidence estimation; consensus decoding; discriminative training, etc.

N-Best List Rescoring

- For exotic models, even lattice rescoring may be too slow.
- Easy to generate *N*-best lists from lattices.
  - A* algorithm.

THE DOG ATE MY
THE DIG ATE MY
THE DOG EIGHT MAY
THE DOGGY MAY

- *N*-best lists have other uses.
  - *e.g.*, confidence estimation; displaying alternatives; etc.

Discussion: A Tale of Two Decoding Styles

- Approach 1: Dynamic graph expansion (since late 1980's).
  - Can handle more complex language models.
  - Decoders are incredibly complex beasts.
  - *e.g.*, cross-word CD expansion without FST's.
  - Graph optimization difficult.
- Approach 2: Static graph expansion (AT&T, late 1990's).
  - Enabled by optimization algorithms for WFSM's.
  - Much cleaner way of looking at everything!
  - FSM toolkits/libraries can do a lot of work for you.
  - Static graph expansion is complex and can be slow.
  - Decoding is relatively simple.

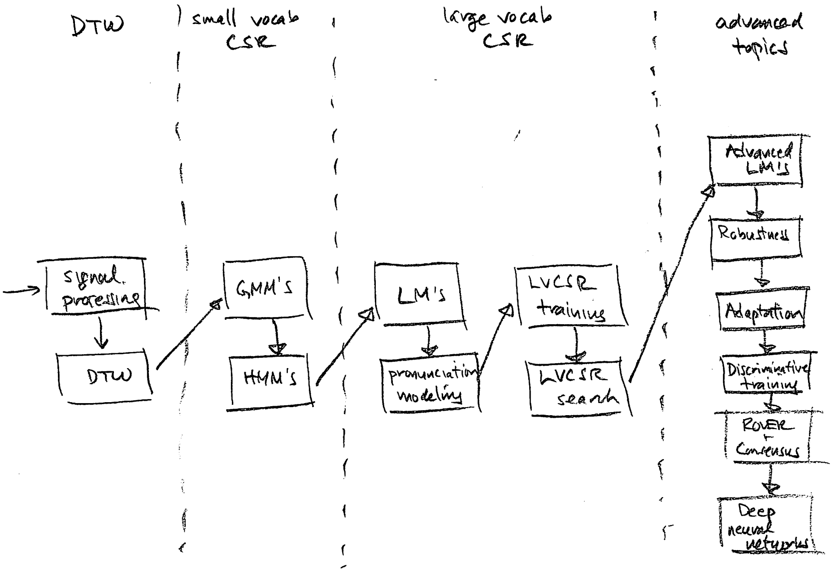
Static or Dynamic? Two-Pass?

- If speed is priority?
- If flexibility is priority?
  - *e.g.*, update LM vocabulary every night.
- If need gigantic language model?
- If latency is priority?
  - What can't we use?
- If accuracy is priority (all the time in the world)?
- If doing cutting-edge research?

References

- F. Pereira and M. Riley, "Speech Recognition by Composition of Weighted Finite Automata", *Finite-State Language Processing*, MIT Press, pp. 431–453, 1997.
- M. Mohri, F. Pereira, M. Riley, "Weighted finite-state transducers in speech recognition", Computer Speech and Language, vol. 16, pp. 69–88, 2002.
- A. Stolcke, "Entropy-based pruning of Backoff Language Models", Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop, pp. 270–274, 1998.

Road Map



DTW
small vocab
CSR
large vocab
CSR
advanced
topics
signal
processing
DTW
GMM's
HMM's
LM's
pronunciation
modeling
LVCSR
training
LVCSR
search
Advanced
LM's
Robustness
Adaptation
Discriminative
training
ROVER
+
Consensus
Deep
neural
networks

Course Feedback

- Was this lecture mostly clear or unclear?
- What was the muddiest topic?
- Other feedback (pace, content, atmosphere, etc.).