Lecture 9

LVCSR Decoding (cont'd) and Robustness

Michael Picheny, Bhuvana Ramabhadran, Stanley F. Chen

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

19 November 2012

Part I

LVCSR Decoding (cont'd)

What Were We Talking About Again?

- Large-vocabulary continuous speech recognition (LVCSR).
- Decoding.
  - How to select best word sequence . . .
  - Given audio sample.
- The basic recipe.
  - Convert LM to giant HMM (*i.e.*, *decoding graph*).
  - Run Viterbi.

What's the Problem?

- Context-dependent graph expansion is complicated.
- Decoding graphs way too big.
- Decoding way too slow.

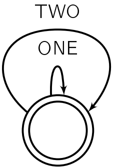
Where Are We?

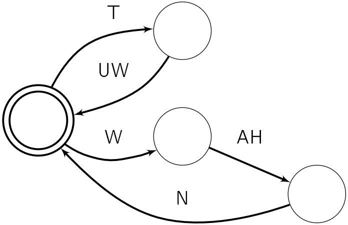
1. Graph Expansion and Finite-State Machines
2. Shrinking the Language Model
3. Graph Optimization
4. Run-time Optimizations
5. Other Decoding Paradigms

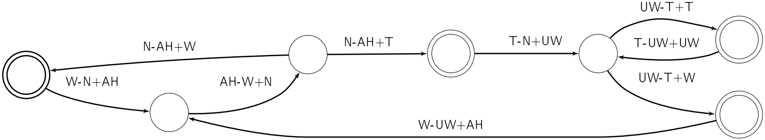
Review: Graph Expansion

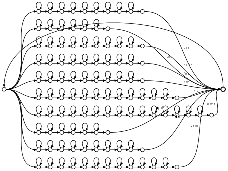
- Start with (n-gram) LM expressed as HMM.
  - Repeatedly expand to lower-level HMM’s.
- This is tricky.
  - Especially expanding from CI to CD phones.
- Natural framework for rewriting graphs:
  - Finite-state acceptors and transducers.

Outline of Graph Expansion



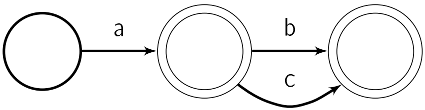




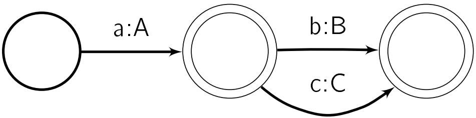


Finite-State Acceptors and Transducers

- FSA represents list of strings.
  - *e.g.*, *a*, *ab*, *ac*.

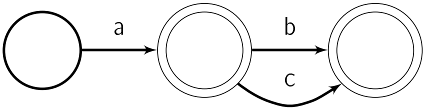


- FST represents list of (*input*, *output*) string pairs:
  - *e.g.*, (*a*, *A*), (*ab*, *AB*), (*ac*, *AC*).

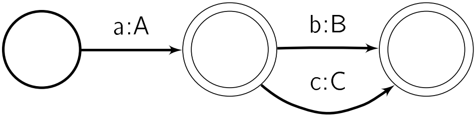


Review: Composition

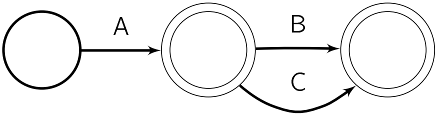
- *A* has meaning: *a*, *ab*, *ac*.



- *T* has meaning: (*a*, *A*), (*ab*, *AB*), (*ac*, *AC*).



- $A \circ T$ has meaning: *A*, *AB*, *AC*.



Composition

- FST's can express wide range of string transformations.
  - 1:1 transformations (*e.g.*, word to baseform).
  - 1:many transformations (*e.g.*, multiple baseforms).
  - 1:0 tranformations (*e.g.*, filter bad language).
- Composition applies to all strings in FSA simultaneously!
- Simple and efficient to compute!

A View of Graph Expansion

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

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

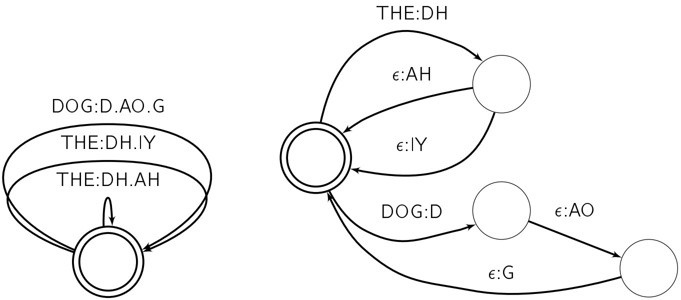
- How to design transducers?

Context-Independent Transformations

- Rewrite string same way independent of context.
  - *e.g.*, word to phones (TWO $\Rightarrow$ `T UW`).
- Create single state.
- Make loop arcs with appropriate input and output.
  - Create extra states/arcs so only one token per arc.
- Don't forget identity transformations!
  - Strings that aren't accepted are discarded.

Example: Mapping Words To Phones

THE	DH AH
THE	DH IY
DOG	D AO G

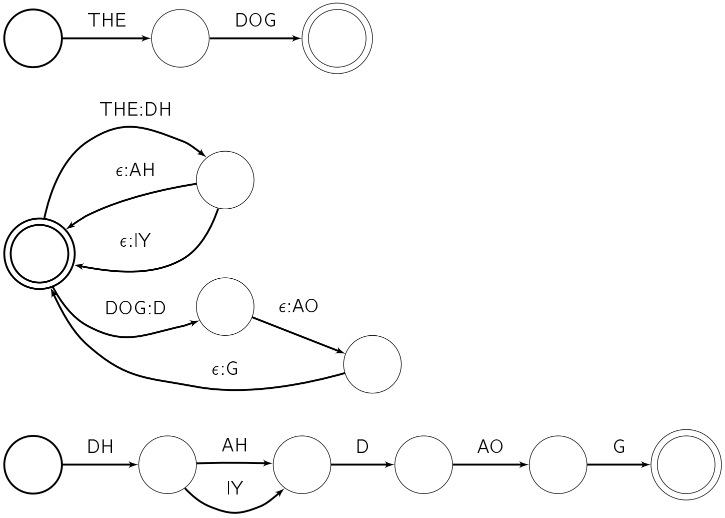


Example: Mapping Words To Phones

A

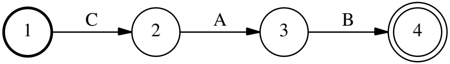
T

$A \circ T$

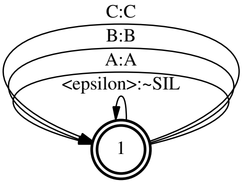


Example: Inserting Optional Silences

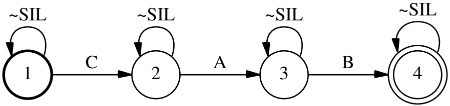
A



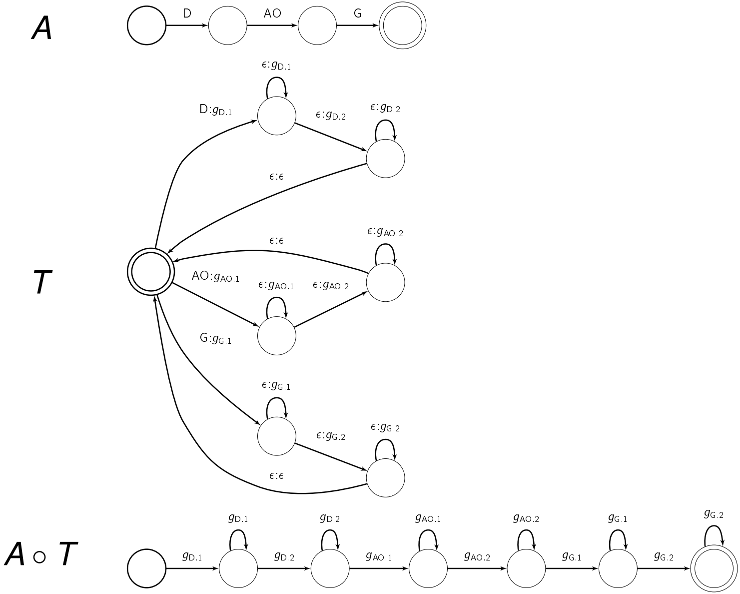
T



$A \circ T$

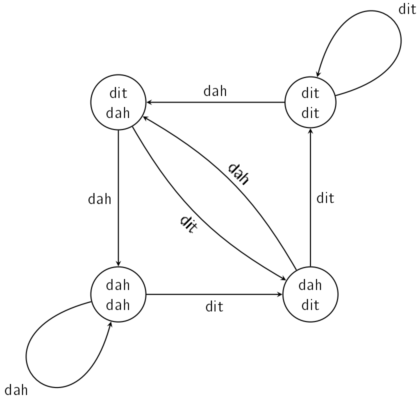


Example: Rewriting CI Phones as HMM's



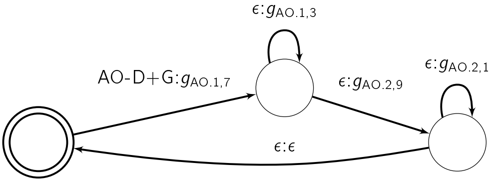
Context-Dependent Transformations

- Rewrite string different ways depending on context.
  - *e.g.*, CI phone to CD phone (`L` ⇒ `L-S+IH`).
- Create one state per "context".
  - *e.g.*, trigram model FSA has state per bigram history.

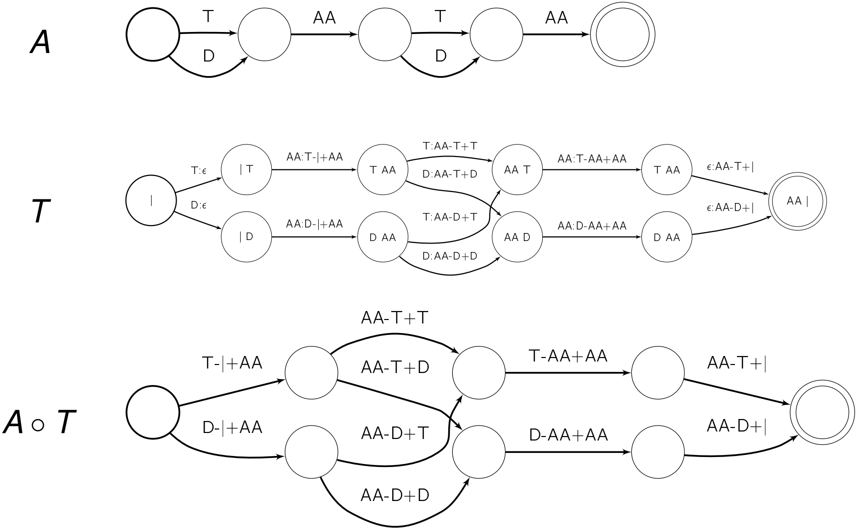


How to Express CD Expansion via FST's?

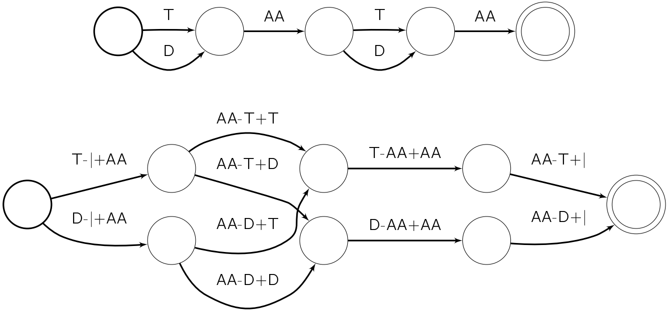
- Step 1: Rewrite each phone as triphone ($\texttt{L} \Rightarrow \texttt{L-S+IH}$).
  - Need to know identity of phone to right!?
  - Idea: delay output of each phone by one arc.
  - State encodes last two phones, like trigram model.
- Step 2: Rewrite each triphone as CD HMM.
  - Compute HMM for each triphone using dcs tree.
  - This transformation is context-independent.



How to Express CD Expansion via FST's?



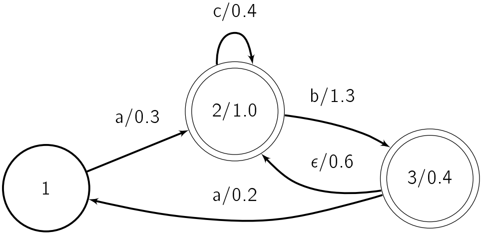
How to Express CD Expansion via FST's?



- 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.)

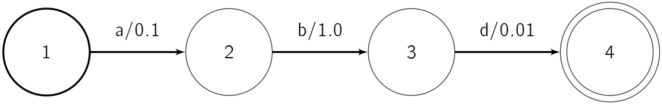
What About Those Probability Thingies?

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

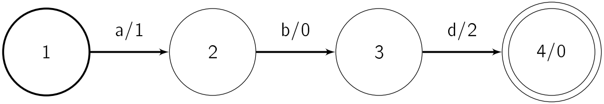


What Is A Cost?

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

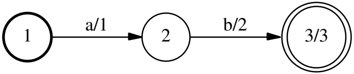


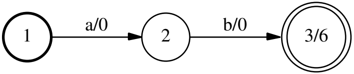
- WFSM's have negative log probs on arcs.
  - Cost of path is sum of arc costs plus final cost.



What Does a Weighted FSA *Mean*?

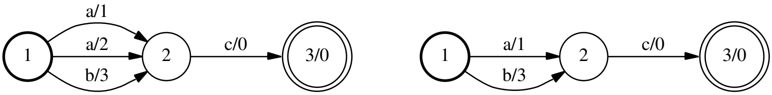
- The (possibly infinite) list of strings it accepts . . .
  - And for each string, a cost.
- Things that *don't* affect meaning.
  - How costs or labels distributed along path.
  - Invalid paths.
- Are these equivalent?





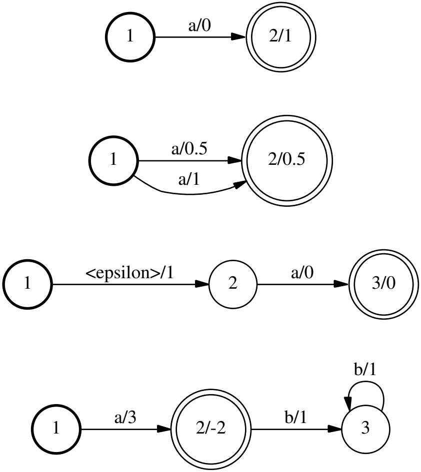
What If Two Paths With Same String?

- How to compute cost for this string?
- Use “min” operator to compute combined cost?
  - Combine paths with same labels; retain meaning.
  - Result of Viterbi algorithm unchanged.

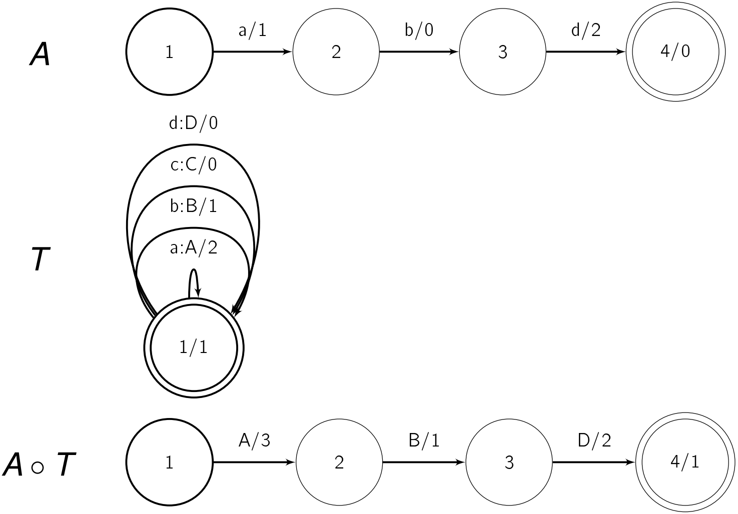


- Operations $(+, \min)$ form a *semiring* (the *tropical* semiring).
  - Other semirings possible.

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):

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

$$P_\omega(\mathbf{x}) = \sum_{\text{paths } A} \prod_{t=1}^{T} p_{a_t} \sum_{\text{comp } j} p_{a_t,j} \prod_{\text{dim } d} \mathcal{N}(x_{t,d}; \mu_{a_t,j,d}, \sigma^2_{a_t,j,d})$$

Recap: FST's and Composition? Awesome!

- Operates on all paths in WFSA (or WFST) simultaneously.
- Rewrites symbols as other symbols.
  - *e.g.*, words as phone sequences (or vice versa).
- Context-dependent rewriting of symbols.
  - *e.g.*, rewrite CI phones as their CD variants.
- Adds in new scores.
  - *e.g.*, language model lattice rescoring.
- Restricts set of allowed paths (intersection).
  - *e.g.*, find all paths containing word ATTACK.
- Or all of above at once.

Weighted FSM's and ASR

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

Discussion

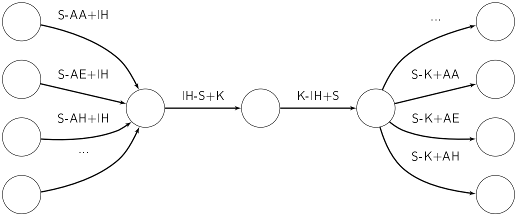
- Don't need to write code?
  - Generate FST's; use FSM toolkit like OpenFST.
- WFSM framework is very flexible.
  - *e.g.*, CD pronunciations at word or phone level.
- Scaling to wider phonetic contexts?
  - Quinphones: $50^5 \approx 300M$ arcs.
  - Given word vocabulary, not all quinphones occur.

Where Are We?

1. Graph Expansion and Finite-State Machines
2. Shrinking the Language Model
3. Graph Optimization
4. Run-time Optimizations
5. Other Decoding Paradigms

The Problem

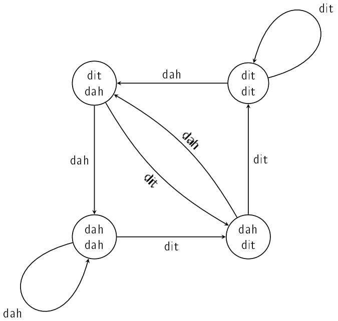
- Naive graph expansion, trigram LM.
  - If $|V| = 50000$, $50000^3 \approx 10^{14}$ word arcs.
  - CI expansion $\Rightarrow$ $\sim$10 states/word.
  - CD expansion $\Rightarrow$ $\gg$10 states/word.



- Graph won't fit in memory.
- Viterbi too slow.
  - Time proportional to number of states (at least).

Compactly Representing *N*-Gram Models

- 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.

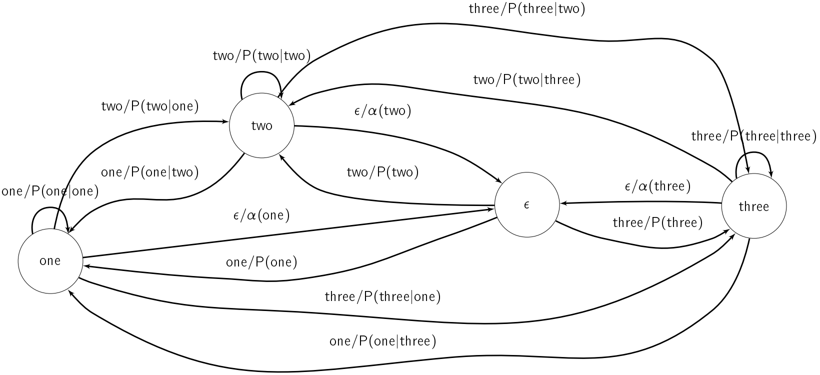
$$P_{\text{smooth}}(w_i|w_{i-1}) = \begin{cases} P_{\text{primary}}(w_i|w_{i-1}) & \text{if count}(w_{i-1}w_i) > 0 \\ \alpha_{w_{i-1}} P_{\text{smooth}}(w_i) & \text{otherwise} \end{cases}$$

- *e.g.*, Witten-Bell smoothing

$$\begin{aligned} P_{\text{WB}}(w_i|w_{i-1}) \quad = \quad & \frac{c_h(w_{i-1})}{c_h(w_{i-1}) + N_{1+}(w_{i-1})} P_{\text{MLE}}(w_i|w_{i-1}) + \\ & \frac{N_{1+}(w_{i-1})}{c_h(w_{i-1}) + N_{1+}(w_{i-1})} P_{\text{WB}}(w_i) \end{aligned}$$

Compactly Representing N-Gram Models

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



Compactly Representing *N*-Gram Models

- By introducing backoff states . . .
  - Only need arcs for *n*-grams with nonzero count.
- Compute probabilities for *n*-grams with zero count . . .
  - By traversing backoff arcs.
- Does this representation introduce any error?
  - Multiple paths with same label sequence?
  - *i.e.*, is this model *hidden*?

Can We Make the LM Even Smaller?

- Sure, just remove some more arcs. Which?
- Count cutoffs.
  - *e.g.*, remove all arcs corresponding to bigrams . . .
  - 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!

Administrivia

- Lab 2, Lab 3 handed back today.
  - `/user1/faculty/stanchen/e6870/lab3_ans/`.
- Lab 4 out tomorrow; due next Thursday, Nov. 29, 11:59pm.
- Make-up lecture: Wednesday, December 5, 4:10–6:40pm?
  - Location: TBA.
- Reading projects.
  - Paper list updated by Wednesday.
  - `http://www.ee.columbia.edu/~stanchen/fall12/e6870/readings/project_f12.html` (same password as readings).
  - Paper selection due next Friday, Nov. 30.
- Non-reading projects.
  - Optional checkpoint next Monday.
  - E-mail to schedule meeting before/after class.

Where Are We?

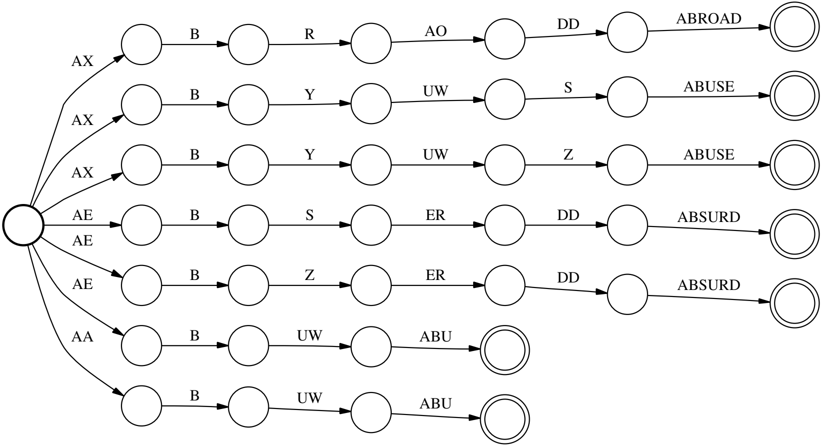
1. Graph Expansion and Finite-State Machines
2. Shrinking the Language Model
3. Graph Optimization
4. Run-time Optimizations
5. Other Decoding Paradigms

Graph Optimization

- Can we modify topology of graph . . .
  - Such that it's smaller (fewer arcs or states) . . .
  - Yet retains same *meaning*.
- Meaning of weighted acceptor:
  - Set of accepted strings; cost of each string.
  - Don't care where costs and labels placed along paths.

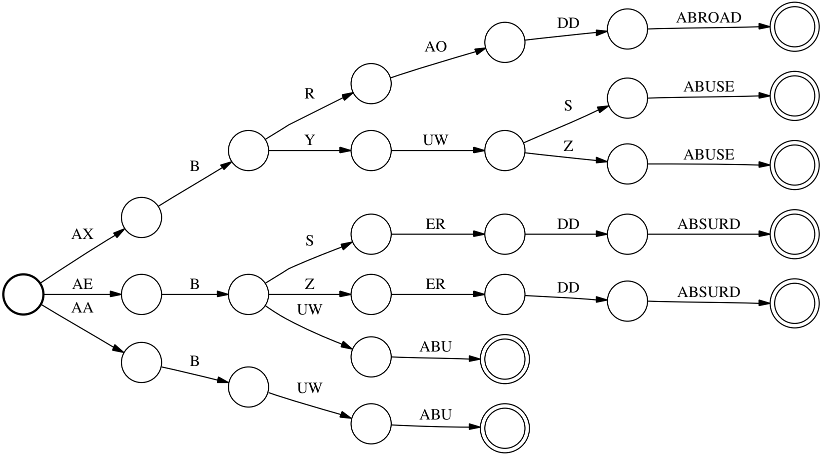
Graph Compaction

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



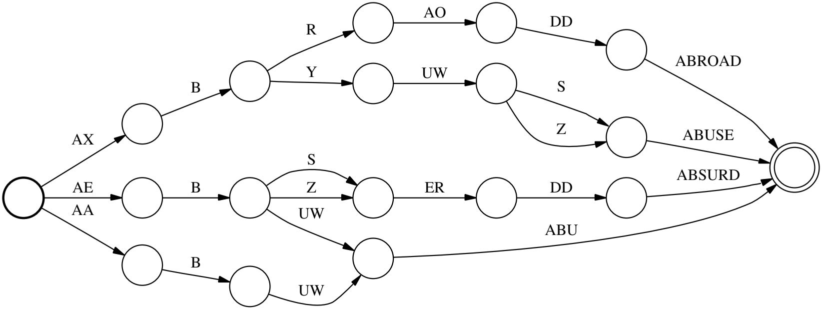
Determinization

- Share common prefixes: 29 states, 28 arcs.



Minimization

- Share common suffixes: 18 states, 23 arcs.

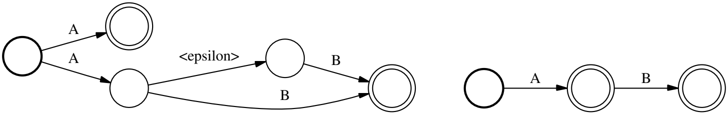


Determinization and Minimization

- By sharing arcs between paths . . .
  - Reduced size of graph by half . . .
  - Without changing meaning!
- *determinization* — prefix sharing.
  - Produce *deterministic* version of FSM.
- *minimization* — suffix sharing.
  - Given deterministic FSM . . .
  - Find equivalent FSM with minimal number of states.

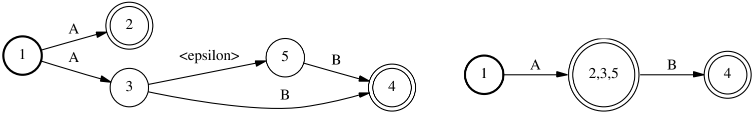
What Is A Deterministic FSM?

- 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.



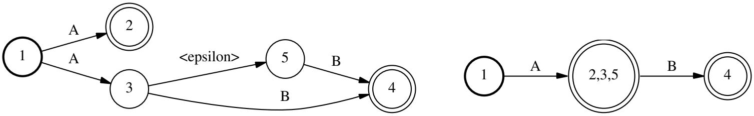
Determinization: The Basic Idea

- For every input label sequence . . .
  - Look at set of states reachable from start state.
- For each unique state set, create state in output FSM.
- Make arcs in logical way.

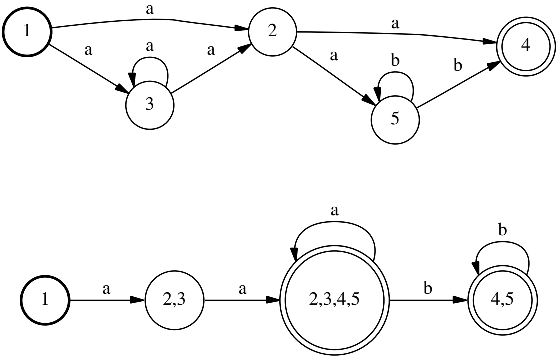


Determinization

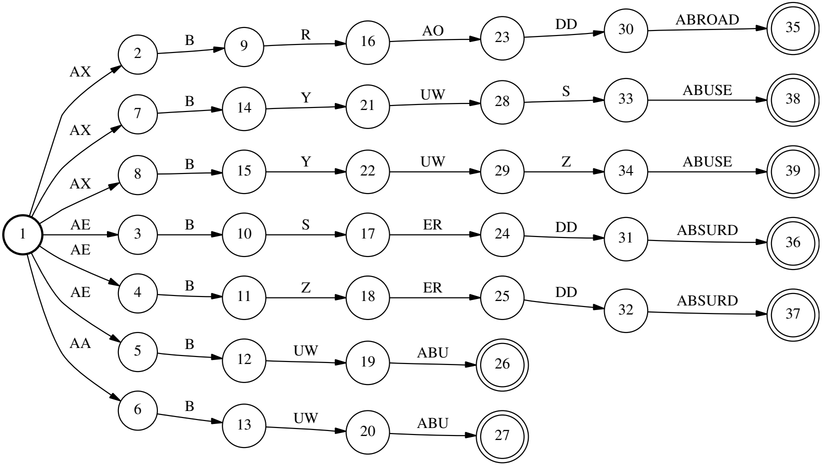
- Start from start state.
- Keep list of state sets not yet expanded.
  - For each, find outgoing arcs, . . .
  - Creating new state sets as needed.
- Must follow ϵ arcs when computing state sets.



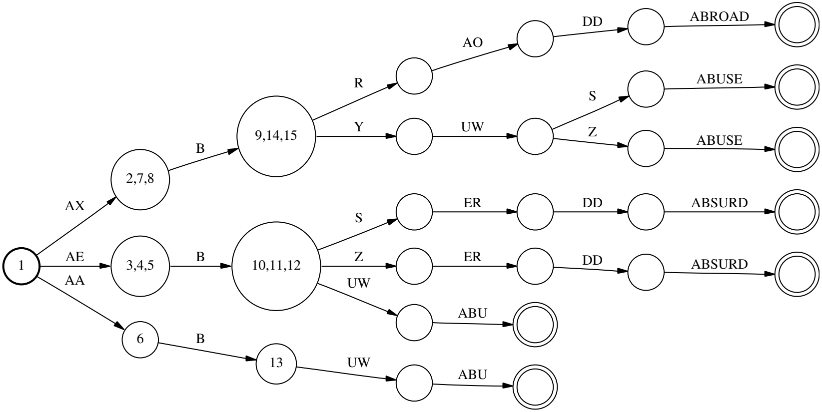
Example 2



Example 3


1
AX
AX
AX
AE
AE
AE
AA
2
7
8
3
4
5
6
B
B
B
B
B
B
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

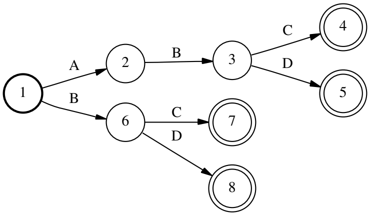


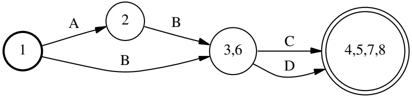
Pop Quiz: Determinization

- 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: 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

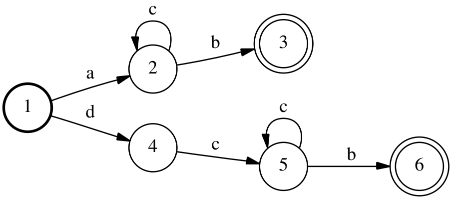
General Minimization: The Basic Idea

- 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.

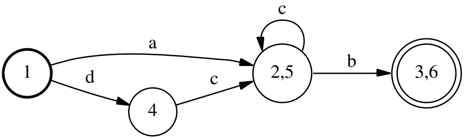
Minimization

- Invariant: if two states are in different partitions . . .
  - They have different follow sets.
  - Converse does not hold.
- First split: final and non-final states.
  - Final states have ϵ in their follow sets.
  - Non-final states do not.
- If two states in same partition have . . .
  - Different number of outgoing arcs or arc labels . . .
  - Or arcs go to different partitions . . .
  - The two states have different follow sets.

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: $\text{min}(\text{det}(L \circ T_{\text{LM}\rightarrow\text{CI}} \circ T_{\text{CI}\rightarrow\text{CD}} \circ T_{\text{CD}\rightarrow\text{GMM}}))$
  - 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. Graph Expansion and Finite-State Machines
2. Shrinking the Language Model
3. Graph Optimization
4. Run-time Optimizations
5. Other Decoding Paradigms

Real-Time Decoding

- Why is this desirable?
- Decoding time for Viterbi algorithm; 10M states in graph.
  - In each frame, loop through every state in graph.
  - 100 frames/sec $\times$ 10M states $\times$ $\ldots$
  - 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.
  - $\Rightarrow$ Pruning!

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.

Pruning

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

$$\omega^* = \operatorname*{arg\,max}_{\omega} P(\omega|\mathbf{x}) = \operatorname*{arg\,max}_{\omega} P(\omega)P_{\omega}(\mathbf{x})$$

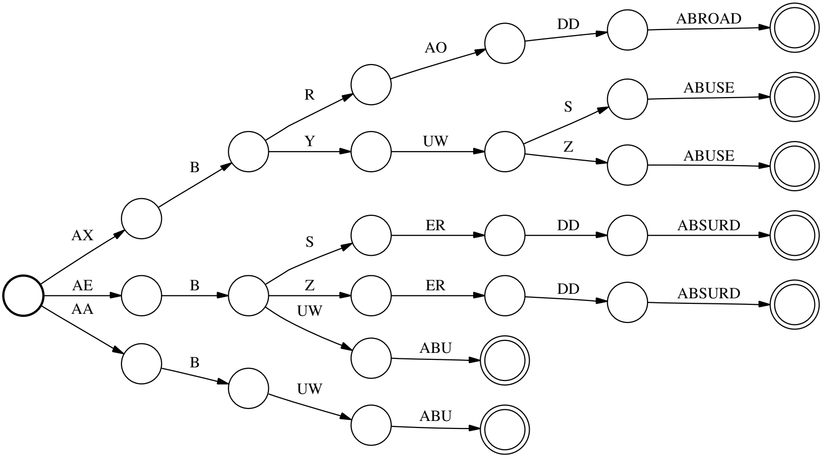
- The goal of *search*:
  - Minimize computation *and* search errors.

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.
  - Why does this make sense? When could this be bad?
- *Rank* or *histogram* pruning.
  - Keep only k highest scoring states.
  - Why does this make sense? When could this be bad?
- Can get best of both worlds?

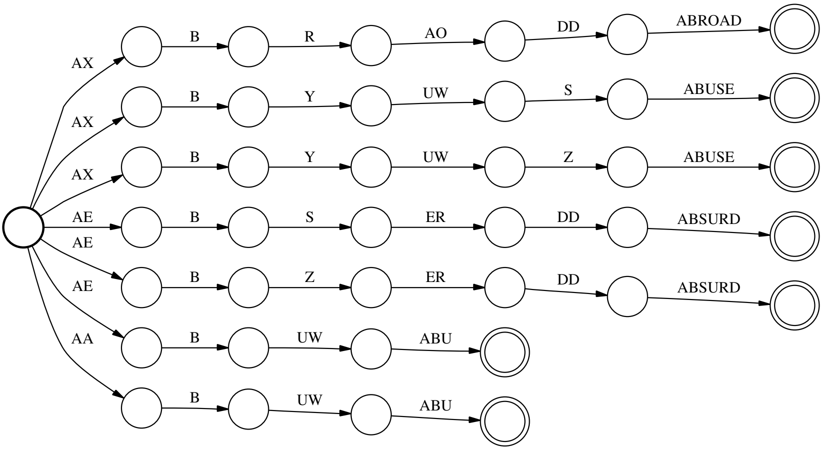
Pruning Visualized

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



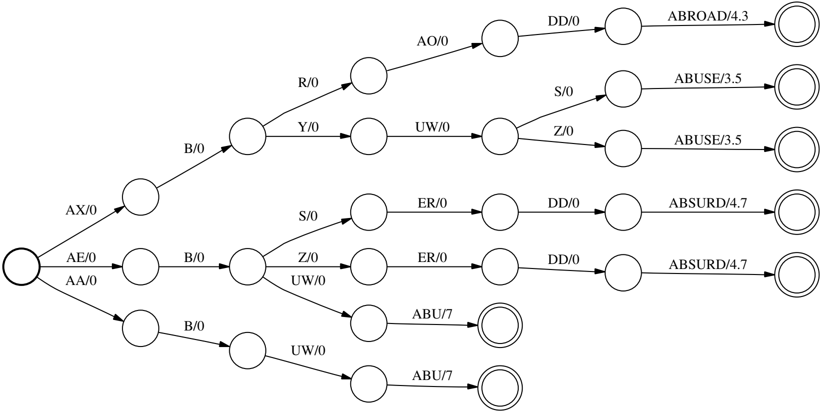
Pruning and Determinization

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



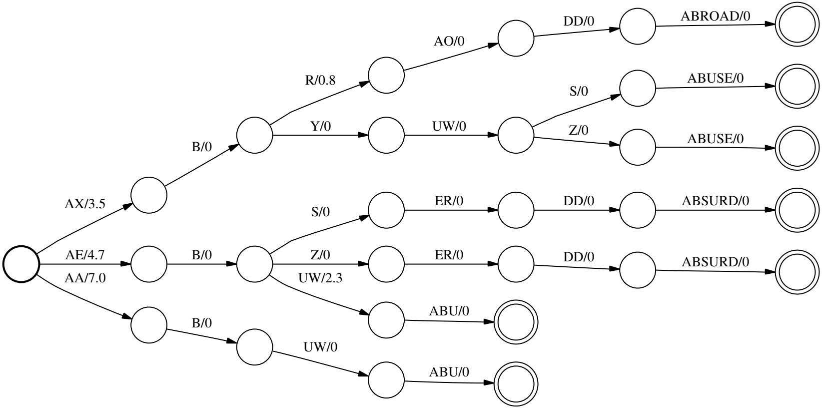
Language Model Lookahead

- 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 ⇔ min LM cost of following words.
- *push* operation does this.



Saving Memory

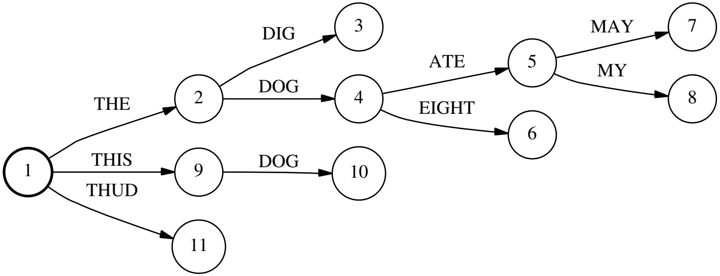
- Naive Viterbi implementation: store whole DP chart.
- If 10M-state decoding graph:
  - 10 second utterance $\Rightarrow$ 1000 frames.
  - 1000 frames $\times$ 10M states = 10 billion cells.
- Each cell holds:
  - Viterbi log prob; backtrace pointer.

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?

Token Passing

- Maintain "word tree":
  - Compact encoding of list of similar word sequences.
  - 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

- 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.
- Memory usage.
  - The biggie: decoding graph (shared memory).

Where Are We?

1. Graph Expansion and Finite-State Machines
2. Shrinking the Language Model
3. Graph Optimization
4. Run-time 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.
  - Can use much larger LM's.

Dynamic Graph Expansion: The Basic Idea

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

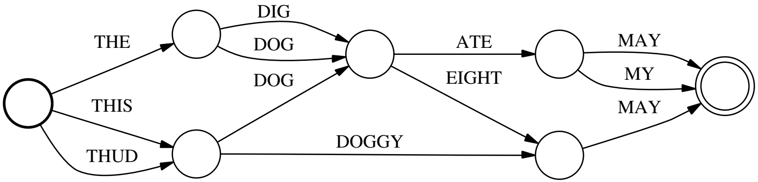
$$
\begin{aligned}
G_{\text{decode}} &= L \circ T_{\text{LM}\rightarrow\text{CI}} \circ T_{\text{CI}\rightarrow\text{CD}} \circ T_{\text{CD}\rightarrow\text{GMM}} \\
&= L \circ (T_{\text{LM}\rightarrow\text{CI}} \circ T_{\text{CI}\rightarrow\text{CD}} \circ T_{\text{CD}\rightarrow\text{GMM}})
\end{aligned}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

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

Where Are We?

- Lectures 1–4: Small vocabulary ASR.
- Lectures 5–8: Large vocabulary ASR.
- Lectures 9–12: Advanced topics.
  - Robustness; adaptation.
  - Advanced language modeling.
  - Discriminative training; ROVER; consensus.
  - Deep Belief Nets (DBN's).
- Lecture 13: Final presentations.

Course Feedback

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