Lecture 8

LVCSR Training and Decoding

Michael Picheny, Bhuvana Ramabhadran, Stanley F. Chen

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

12 November 2012

What We're Talking About Today

- Large-vocabulary continuous speech recognition (LVCSR).
- Acoustic model training.
  - How to estimate parameters, *e.g.*, for GMM's.
  - How to build phonetic decision trees.
- Decoding.
  - How to select best word sequence . . .
  - Given audio sample.

Part I

LVCSR Training

Review

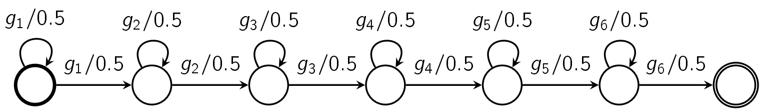
- $\mathbf{x}$ — Observations; sequence of $\sim$40d feature vectors.
- ω — word sequence.
- Fundamental equation of ASR.

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

- $P_{\omega}(\mathbf{x})$ — acoustic model.
  - For word sequence ω, how likely are features $\mathbf{x}$?
- $P(\omega)$ — language model.
  - How likely is word sequence ω?

Review: Acoustic Modeling

- For word sequence ω, construct associated HMM.



- Each $\mathbf{x}$ can be output by many *paths* through HMM.
- Compute $P_\omega(\mathbf{x})$ by summing over path likelihoods.

$$P_\omega(\mathbf{x}) = \sum_{\text{paths } A} P_\omega(\mathbf{x}, A)$$

- Compute path likelihood by ...
  - Multiplying arc and GMM output probs along path.

Acoustic Likelihoods: Small Vocabulary

$$\begin{aligned} P_\omega(\mathbf{x}) &= \sum_{\text{paths } A} P_\omega(\mathbf{x}, A) \\ &= \sum_{\text{paths } A} \prod_{t=1}^{T} p_{a_t} \times P(\vec{x}_t | a_t) \\ &= \sum_{\text{paths } A} \prod_{t=1}^{T} p_{a_t} \sum_{\text{comp } j} p_{a_t,j} \prod_{\text{dim } d} \mathcal{N}(x_{t,d}; \mu_{a_t,j,d}, \sigma^2_{a_t,j,d}) \end{aligned}$$

- p_a — transition probability for arc a.
- $p_{a,j}$ — mixture weight, jth component of GMM on arc a.
- $\mu_{a,j,d}$ — mean, dth dim, jth component, GMM on arc a.
- $\sigma^2_{a,j,d}$ — variance, dth dim, jth component, GMM on arc a.

Acoustic Likelihoods: Large Vocabulary

$$\begin{aligned} P_\omega(\mathbf{x}) &= \sum_{\text{paths } A} P_\omega(\mathbf{x}, A) \\ &= \sum_{\text{paths } A} \prod_{t=1}^{T} p_{a_t} \times P(\vec{x}_t | a_t) \\ &= \sum_{\text{paths } A} \prod_{t=1}^{T} p_{a_t} \sum_{\text{comp } j} p_{a_t,j} \prod_{\text{dim } d} \mathcal{N}(x_{t,d}; \mu_{a_t,j,d}, \sigma^2_{a_t,j,d}) \end{aligned}$$

- p_a — transition probability for arc a.
- $p_{a,j}$ — mixture weight, jth component of GMM on arc a.
- $\mu_{a,j,d}$ — mean, dth dim, jth component, GMM on arc a.
- $\sigma^2_{a,j,d}$ — variance, dth dim, jth component, GMM on arc a.

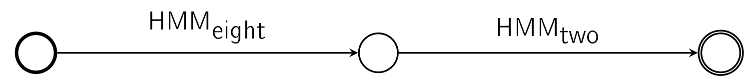
So, What's Different for Large Vocabulary?

- The HMM.

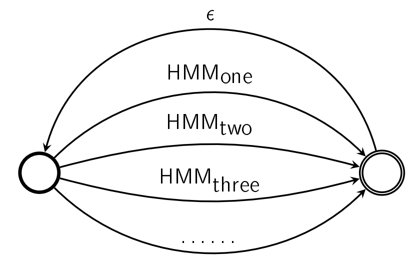
Where Are We?

Review: Building HMM's, Small Vocabulary

- Training.
  - Enumerate possible word sequences given transcript.
  - Replace each word with its HMM; collect FB counts.

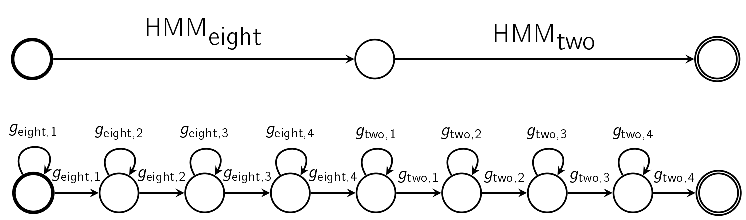


- Decoding.
  - Enumerate possible word sequences.
  - Replace each word with its HMM; run Viterbi.



Example: Word Models (Training HMM)

- One HMM per word (two states per phone, say).
  - Each HMM has own GMM's (one per state).
- *e.g.*, reference transcript: *EIGHT TWO*.
  - EY TD T UW



What's the Problem With Word Models?

- What if want to be able to decode ...
  - Word not in training set, *e.g.*, *REDONKULOUS*?
- Lots of data for some words.
  - Almost no data for others.
- Not scalable to large vocabulary.

Phonetic Modeling

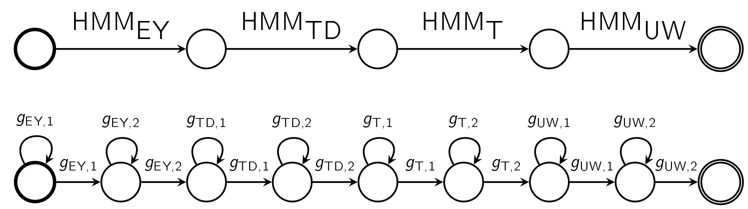
- One HMM per phoneme.
  - Each HMM has own GMM's.
- Need pronunciation or *baseform* for each word.

$$\mathit{TWO} \Rightarrow \mathit{T\ UW}$$
$$\mathit{TEN} \Rightarrow \mathit{T\ EY\ N}$$

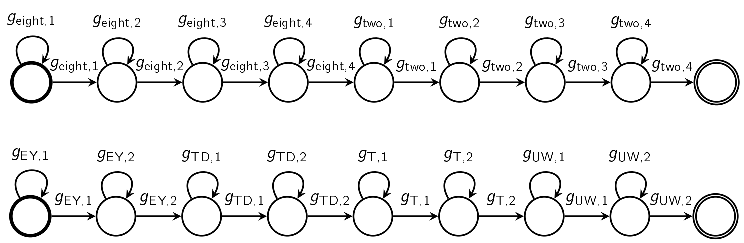
- Concatenate phoneme HMM's to form HMM for word.
  - *i.e.*, share GMM's for phone across all words . . .
  - Containing that phone.
- What if word not in training? No problemo.
- What if phoneme not in training? Unlikely.

Phonetic Modeling

$$\mathit{TWO} \Rightarrow \mathit{T\ UW}$$
$$\mathit{EIGHT} \Rightarrow \mathit{EY\ TD}$$



What's the Difference?



- HMM topology typically doesn't change.
- HMM parameterization changes.

Pop Quiz

- Scenario:
  - 1000 word vocabulary; 50 phonemes.
  - Avg. word length = 5 phones; two states per phoneme.
- Word modeling: one HMM per word.
  - How many GMM's per word on average?
  - How many GMM's in whole system?
- Phonetic modeling: one HMM per phoneme.
  - How many GMM's per phoneme?
  - How many GMM's in whole system?

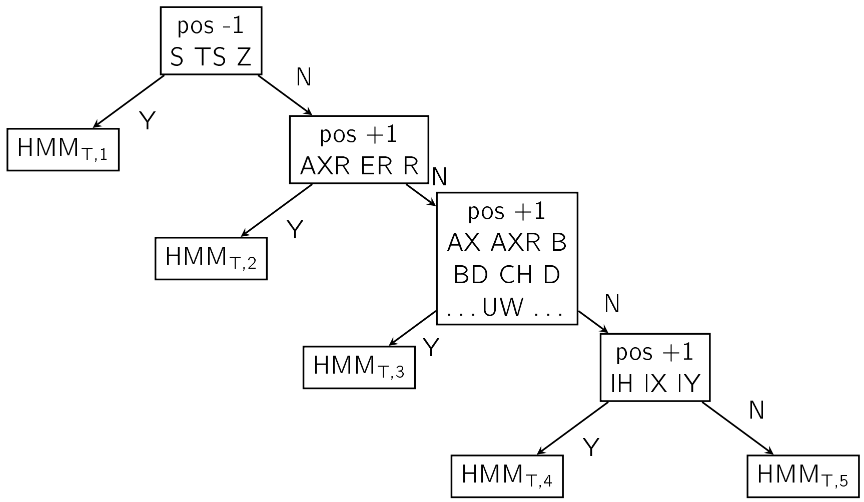
Context-Independent Phonetic Modeling

- Same phoneme HMM independent of phonetic context.
- What's the problem?
  - Is '*L*' in '*S L IH*' and '*IH L Z*' the same?
  - Allophonic variation; coarticulation.
- Symptom: too few GMM's ⇒ underfitting.

Context-Dependent Phonetic Modeling

- Separate HMM for each context of each phoneme?
  - *e.g.*, *triphone* model ⇒ context is ± 1 phone.
  - Separate HMM for *L-S+IH*, *L-IH+Z*, . . .
- What's the problem?
- Solution: cluster triphones.
  - *e.g.*, *L-S+IH*, *L-S+AA*, *L-S+AE*, *L-S+EH*, . . .
  - Separate HMM for each *cluster*.
  - Most popular method: decision trees.

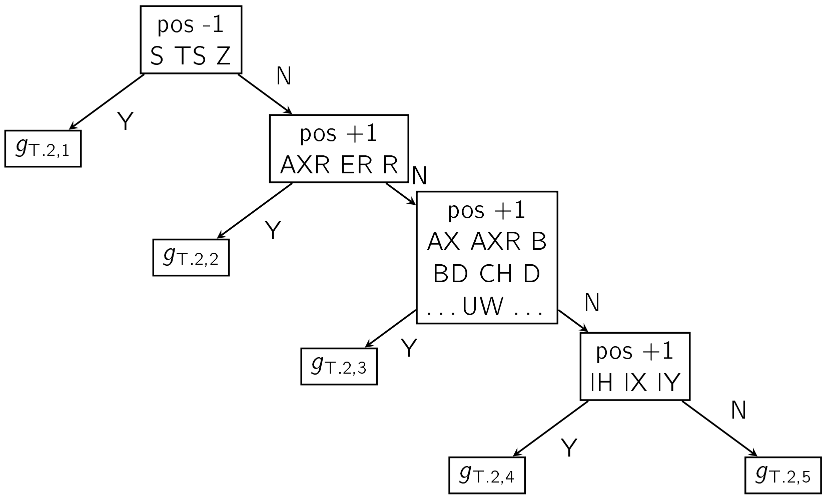
Example: Tree for Phoneme T



How Many Trees?

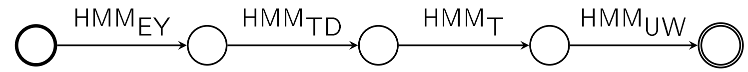
- Which phoneme position affects pronunciation of . . .
  - Beginning of current phoneme the most?
  - What about end of current phoneme?
- Separate decision tree for each phoneme HMM state!
  - If 50 phones, 2 states/phone, how many trees total?
  - For each tree, one GMM per leaf.
- HMM topology fixed.
  - Choose GMM to use at each position . . .
  - By finding leaf in corresponding tree.

Example: Tree for Phoneme T, State 2

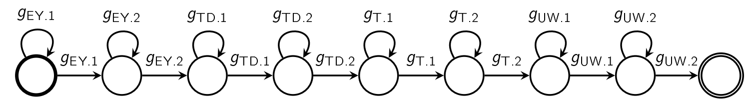


Context-Dependent Phonetic Modeling

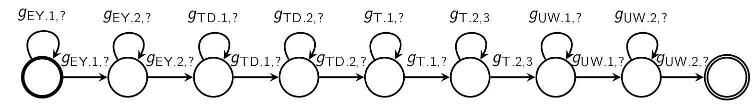
- Start with phoneme sequence.



- Substitute in HMM topology for each phoneme.



- Select GMM for each state using associated tree.



Pop Quiz

- Scenario:
  - 1000 word vocabulary; 50 phonemes.
  - Avg. word length = 5 phones; two states per phoneme.
  - Each decision tree contains 100 leaves on average.
- Word modeling: one HMM per word.
  - How many GMM's per word on average? 10.
  - How many GMM's in whole system? 10,000.
- Phonetic modeling, CI: one HMM per phoneme.
  - How many GMM's per phoneme? 2.
  - How many GMM's in whole system? 100.
- Phonetic modeling, CD: many HMM's per phoneme.
  - How many GMM's per phoneme?
  - How many GMM's in whole system?

Size Matters

- Typical model sizes:

type	HMM	GMM's/ state	GMM's	Gaussians
word	per word	1	10–500	100–10k
CI phone	per phone	1	∼150	1k–3k
CD phone	per phone	1–200	1k–10k	10k–300k

- 40d feature vectors ⇒ 80 parameters/Gaussian.
- Big models can have tens of millions of parameters.

Recap

- Word modeling doesn't scale.
  - Don't share data between words.
  - Some words have lots of data; other very little.
  - Can't model coarticulation across words.
- Phonetic modeling scales.
  - Share data between words; parameter tying.
  - Every phoneme has lots of data . . .
  - But some lots more than others.
- Context-dependent phonetic modeling.
  - Models coarticulation, including cross-word.
  - More data $\Rightarrow$ more leaves $\Rightarrow$ more parameters.
  - Can spread data evenly across GMM's.

Discussion

- CD phonetic modeling with decision trees.
  - State of the art since early 1990's.
  - No serious challenger on horizon?
  - *triphone* model — ± 1 phones of context.
  - *quinphone* model — ± 2 phones of context.
  - Longer context makes decoding much harder!
- Basic issue: parameter tying.
  - Each state for each phoneme has own decision tree.
  - Each leaf in each decision tree has own GMM.
  - Share leaf GMM across all words containing leaf.
  - What are other possible schemes?

Where Are We?

Training $\Leftrightarrow$ Parameter Estimation

- Likelihood of training data is function of parameter values.
  - Transition probabilities.
  - GMM's: mixture weights; means and variances.
- Find parameter values to maximize likelihood.
- Tool: Forward-Backward algorithm.
  - Given initial values, iteratively adjust parameters . . .
  - To improve likelihood.
  - *i.e.*, find closest local maximum to start.

Small Vocabulary Training — Lab 2

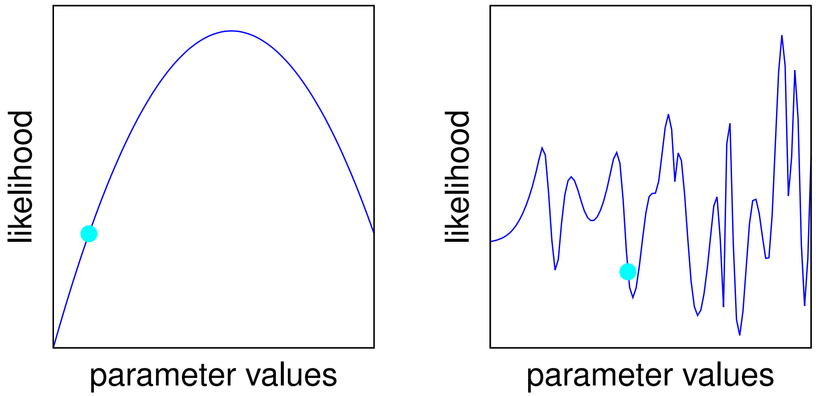
- Phase 1: Flat start.
  - Initialize all Gaussian means to 0, variances to 1.
- Phase 2: Run Forward-Backward algorithm to convergence.
- Phase 3: Profit!

Large Vocabulary Training

- What's changed?
  - Lab 2: $<$2500 parameters.
  - Large vocabulary: up to 10M+ parameters.
- Realistically, can't do simple hill-climbing search . . .
  - On 10M+ parameters and find good local maximum.
  - It's a miracle it works with 2500 parameters.

Hill Climbing and Local Maxima

- FB finds "nearest" maximum to initial parameters.
  - With bad starting point, final model will be garbage.
- How to find good starting point?



Where Do Local Maxima Come From?

- ML estimation for non-hidden models is easy.
  - *e.g.*, non-hidden HMM's; Gaussians; multinomials.
  - Count and normalize; no search necessary.
- Problem must be hidden variables!

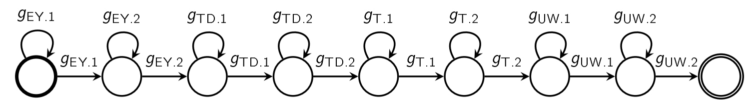
What Are The Hidden Variables?

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

- Look for sums or max's.
- Path through HMM $\Rightarrow$ which GMM/state at each frame.
- Which component in each GMM at each frame.

Hidden Variables and Local Maxima

- Assume each GMM has single component $\Rightarrow$ not hidden.
- Let's assign values to every hidden variable . . .
  - In whole training set.
  - *i.e.*, which GMM generates each frame.



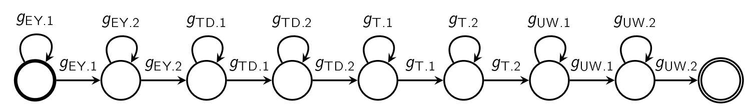
frame	0	1	2	3	4	5	. . .
GMM	EY.1	EY.1	EY.2	EY.2	EY.2	TD.1	. . .

- Call hidden assignment over whole corpus an *alignment*.

Alignments and Parameter Initialization

- Fixing alignment $\Rightarrow$ making corpus non-hidden.
  - Easy to do ML estimation of parameters.
  - Like Viterbi-style training in Lab 2.
  - *i.e.*, can use alignment to initialize parameters.
- Data used to train given GMM comes from . . .
  - All frames aligned to that GMM.
- If seed parameters using "bad" alignment . . .
  - Wrong data used to train GMM's.
  - Parameters near bad maximum?
- If seed parameters using "good" alignment . . .
  - Right data used to train GMM's.
  - Parameters near good maximum?

Example: Good and Bad Alignments



- Good alignment — matches "truth".
  - GMM models what it's supposed to be modeling.
  - *e.g.*, GMM associated with first state of *TD-EY+T* . . .
  - Aligns to initial frames of '*TD*' in this context.

frame	0	1	2	3	4	5	. . .
truth	EY.1	EY.1	EY.2	EY.2	EY.2	TD.1	. . .
hyp	EY.1	EY.1	EY.2	EY.2	EY.2	TD.1	. . .

- Bad alignment — doesn't match "truth".

frame	0	1	2	3	4	5	. . .
truth	EY.1	EY.1	EY.2	EY.2	EY.2	TD.1	. . .
hyp	EY.1	EY.2	EY.2	TD.1	TD.1	TD.2	. . .

Parameter Initialization

- Key to finding good starting point for FB:
  - Need good alignment to seed parameters!
- Point: if have existing "good" model . . .
  - Use model to compute (Viterbi) alignment.
  - Use alignment to bootstrap another model.
  - Repeat to build more and more complex models!
- Where to get first "good" model?
  - Where does FB with flat start actually work!?
- Build lots of incrementally more complex models . . .
  - Or go straight from initial model to final model?

The Basic Plan

- Step 1: Build CI model with 1 Gaussian/GMM.
  - Know flat start + FB works!
- Step 2: Build CI model with 2 Gaussians/GMM.
  - Seed using alignment from last system; run FB.

. . . . . .
. . . . . .
. . . . . .

- Step k: Build CD model with 128 Gaussians/GMM.
  - Seed using alignment from last system; run FB.

Ways to Seed Next Model From Last One

- Via alignment.
  - Do Viterbi-style training for next model . . .
  - Using Viterbi alignment computed using last model.

frame	0	1	2	3	4	5	. . .
GMM	EY.1	EY.1	EY.2	EY.2	EY.2	TD.1	. . .

- Via parameters.
  - Seed parameters of next model so . . .
  - Viterbi alignment is same (or close) as for last model.
  - *e.g.*, GMM splitting (clone each Gaussian, perturb).
  - *e.g.*, CI $\Rightarrow$ CD GMM's (clone each CI GMM).

Recap

- For models with millions of parameters . . .
  - Flat start and FB just doesn't cut it.
- Local maxima due to hidden variables.
  - *i.e.*, space of possible alignments.
- If have good alignment . . .
  - Can initialize parameters so near good maximum.
- Key idea: use simple models to bootstrap . . .
  - Incrementally more complex models.
- More gory details to follow.

Where Are We?

Overview of Training Process

- Start: CI, GMM's contain single component.
- End: CD, GMM's contain 128 components, say.
- How to get here from there?
  - More than one way.
- Let's go through one recipe, start to finish.

Step 0: Prerequisites

- Data.
  - Utterances with transcripts.
  - Pronunciation/baseform dictionary.
  - Questions to ask in phonetic decision tree.
- Decisions.
  - For each phoneme, HMM topology/size.
  - Number of components in GMM's.
- Period.

The Pronunciation Dictionary

- Need pronunciation of *every* word in training data.
  - Without pronunciation, can't build HMM for word.
- Words may have multiple pronunciations.

```
THE(01)   DH  AH
THE(02)   DH  IY
```

- Where to get baseforms for new words?
  - Ask a linguist? (We fired them.)
  - Where else?

Step 1: CI, 1 component/GMM

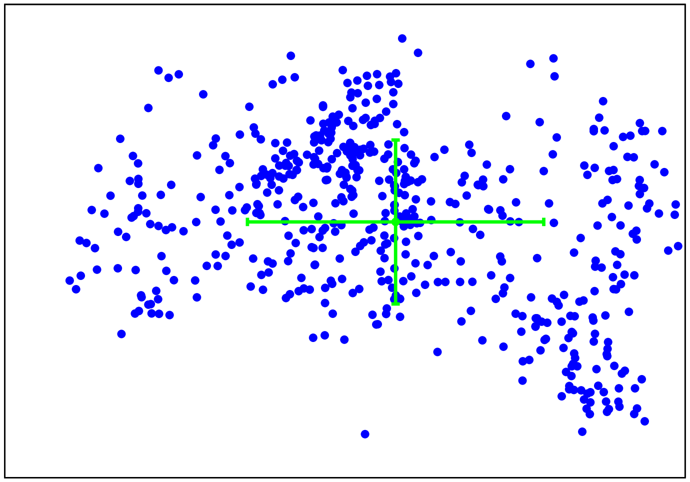
- Flat start.
  - Transition probabilities, mixture weights uniform.
  - Gaussian means 0, variances 1.
- Run FB to convergence (Lab 2).
- Before: alignments are garbage.
- After: alignments are reasonable (but flawed).

Step 2: CI, 32 components/GMM

- Split Gaussians $\Rightarrow$ 2 components/GMM.
  - Run bunch of iterations of FB.
- Split Gaussians $\Rightarrow$ 4 components/GMM.
  - Run bunch of iterations of FB.
- Split Gaussians $\Rightarrow$ 8 components/GMM.
  - Run bunch of iterations of FB.
- Split Gaussians $\Rightarrow$ 16 components/GMM.
  - Run bunch of iterations of FB.

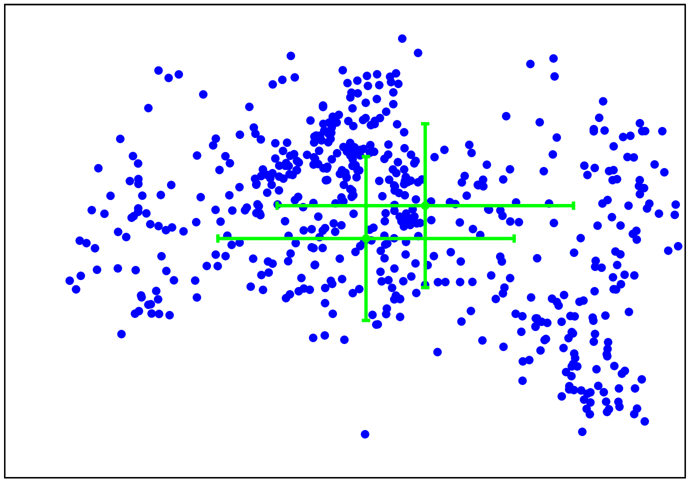
Example: Gaussian Splitting

- Train single Gaussian via Forward-Backward.



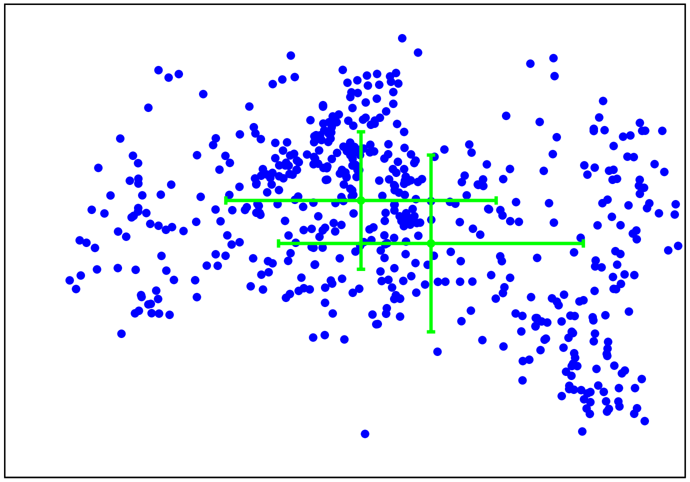
Example: Gaussian Splitting

- Split each Gaussian in two ($\pm 0.2 \times \vec{\sigma}$)



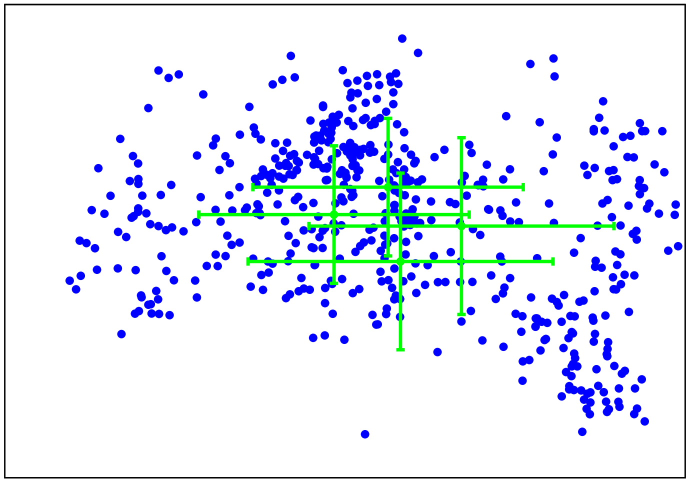
Example: Gaussian Splitting

- Run FB for a few iterations.



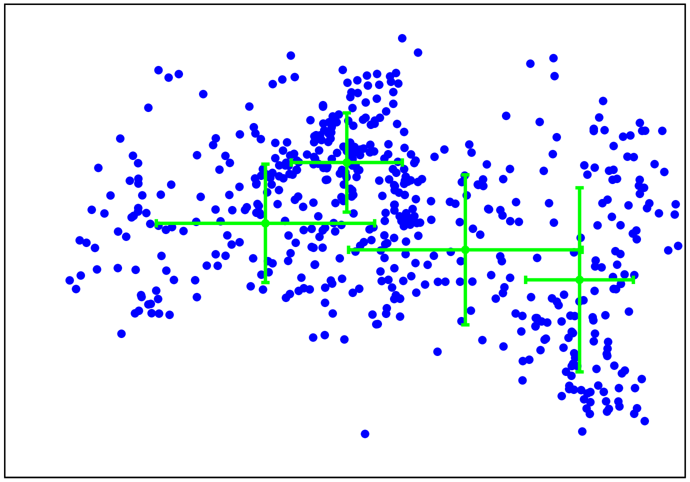
Example: Gaussian Splitting

- Split each Gaussian in two ($\pm 0.2 \times \vec{\sigma}$)



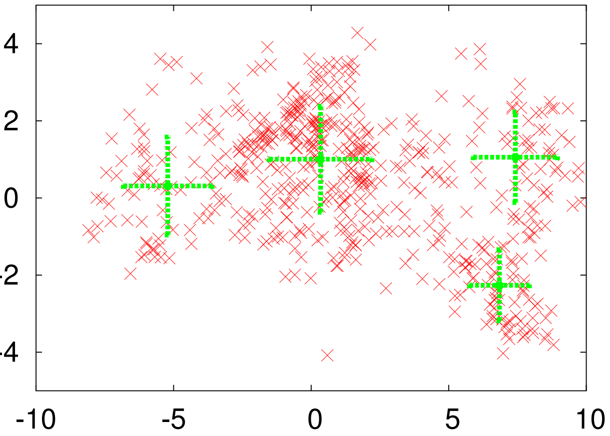
Example: Gaussian Splitting

- Run FB for a few iterations.

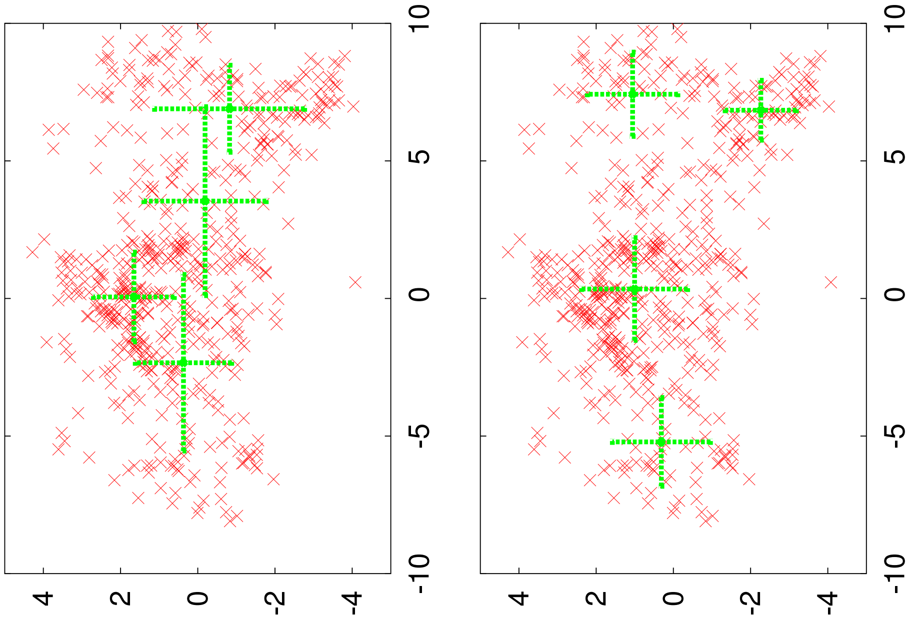


There is also *k*-Means

- Use centers as means of Gaussians; train.



The Final Mixtures, Splitting *vs.* *k*-Means



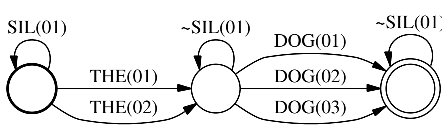
Step 3: Select Pronunciation Variants

- Reference transcript doesn't tell you everything.
- Missing silence, filled pauses (*e.g.*, *UH*).
- Doesn't tell you which pronunciation . . .
  - For words with multiple pronunciations.
  - *e.g.*, whether *THE* pronounced '*DH AH*' or '*DH IY*'.

```
THE(01)    DH  AH
THE(02)    DH  IY
```


Handling All Possible Alternatives

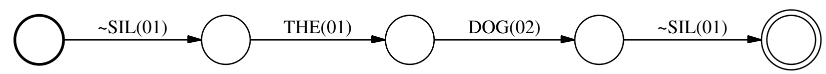
- In theory, optional silence, multiple pronunciations . . .
  - No problem! Just build appropriate HMM.
  - Consider all possible paths over whole training process.



- In practice, painful.
  - Expensive computationally.
  - Building training HMM with CD models tricky.

What To Do?

- Solution: nail down "exact" transcript.



- Once model sufficiently good, compute Viterbi path.
  - Identify pronunciations (and silences) along path.
- Fix "exact" transcript for remainder of training.
  - Or recompute periodically.

Step 3: Select Pronunciation Variants

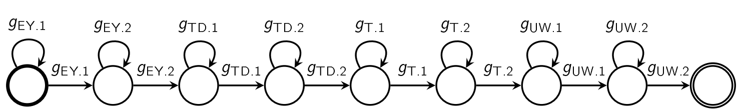
- Run Viterbi algorithm on training set.
  - Compute “exact” transcript for each utterance.
- Run bunch of iterations of FB.

Step 4: Building Phonetic Decision Trees

- Goal: build phonetic decision tree ...
  - For each state in each phone HMM ($\sim$150 total).
  - *e.g.*, `AA.1`, `AA.2`, `AA.3`, `AE.1`, ...
- What do we need?
  - Data aligned to each phone HMM state.
  - List of candidate questions.

Training Data for Decision Trees

- Run Viterbi algorithm.
  - For each frame, identify which feature vector, ...
  - Which GMM/HMM state, and phonetic context.



frame	0	1	2	3	4	5	...
GMM	EY.1	EY.1	EY.2	EY.2	EY.2	TD.1	...

- *e.g.*, feature vector $\mathbf{x}_5$ used to train tree for `TD.1`.
  - (Triphone) context is `-EY+T`.
  - Data for tree is list of triples $(\vec{x}, p_L, p_R)$; *e.g.*, $(\mathbf{x}_5, \texttt{EY}, \texttt{T})$.

Building a (Triphone) Tree

- Input: list of triples $(\vec{x}_i, p_{L,i}, p_{R,i})$.
- At each node on frontier of tree:
  - Choose question of form ...
  - “*Does phone in position j belong to set q?*” ...
  - Optimizing $\prod_i P(\vec{x}_i|\text{leaf}(p_{L,i}, p_{R,i}))$ ...
  - Where each leaf distribution is single Gaussian.
- Can efficiently build whole level of tree in single pass.
- See Lecture 6 slides and readings for gory details.

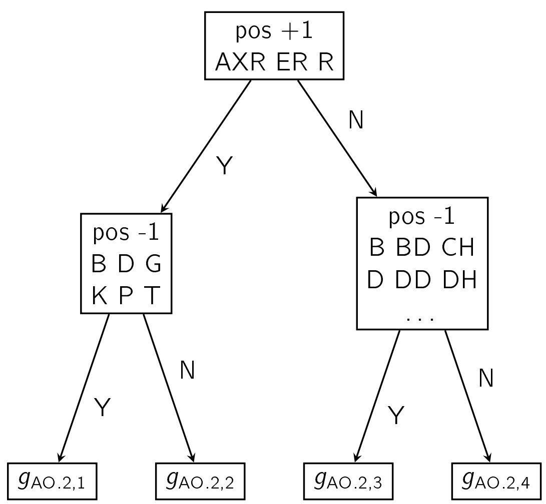
The List of Candidate Questions

- Created by linguist many decades ago.
  - Passed down from mother to daughter, father to son.
- Corresponds to phonetic concepts.
  - *e.g.*, vowel? dipthong? fricative? nasal? etc.
- Each question represented as set of phones.
  - Does phoneme belong to set of not?

Example Questions

- AA
- AE
- ...
- ZH
- AO OY
- AX IH
- CH JH
- DH V
- ER R
- F TH
- IH IY
- IY Y
- L W
- OW UW
- SH ZH
- S Z
- AE EH EY
- B D G
- F HH TH
- K P T
- M N NG
- S TS Z
- AH AO AX EY
- CH JH SH ZH
- DH F TH V
- ...

Example Output



Step 4: Building Phonetic Decision Trees

- Build phonetic decision tree for each phone state.
- Before: one (CI) GMM per phone state.
  - After: one (CD) GMM per leaf for each phone state.
  - Seed CD GMM's by cloning original CI GMM.
- Initially, same Viterbi alignment as CI model.
  - In computing likelihood, replace CI with CD GMM ...
  - But these are identical.
- Run bunch of iterations of FB.

Step 5: CD, 128 components/GMM

- Split Gaussians ⇒ 32 components/GMM.
  - Run bunch of iterations of FB.
- Split Gaussians ⇒ 64 components/GMM.
  - Run bunch of iterations of FB.
- Split Gaussians ⇒ 128 components/GMM.
  - Run bunch of iterations of FB.

Recap

- Step 0: Collect data.
  - Make baseforms for all words in reference transcripts.
- Step 1: Build CI, 1 component/GMM model from flat start.
- Step 2: Build CI, many component GMM model.
  - Repeated Gaussian splitting.
- Step 3: Find “exact” transcripts, pronunciation variants.
  - Viterbi algorithm.
- Step 4: Build phonetic decision tree.
  - From alignment created by CI model.
- Step 5: Build CD, many component GMM model.
  - Repeated Gaussian splitting.

Discussion

- One of many possible recipes.
- Training is complicated, multi-step process.
- Motifs.
  - Seed complex model using simpler model.
  - Run lots of Forward-Backward.

Where Are We?

LVCSR Training Doesn't Require Much

- Data.
  - Utterances with transcripts.
  - Pronunciation/baseform dictionary.
  - Questions to ask in phonetic decision tree.
- Algorithms.
  - Viterbi; Forward-Backward.
  - Decision-tree building.
  - Almost same as in small vocabulary.

Training Is an Art

- Hidden model training fraught with local maxima.
- Seed more complex models with simpler models.
  - Incrementally improve alignments; avoid bad maxima.
- Recipes developed over decades.
  - Discovered via sweat and tears.
- No one believes these find global maxima.
  - How well recipe works depends on data?

Speeding Up Training

- Requires many, many iterations of Forward-Backward.
- Full Forward-Backward training.
  - Compute posterior of each alignment.
  - Collect counts over all possible alignments.
- Viterbi-style training.
  - Pick single alignment, *e.g.*, using Viterbi.
  - Collect counts over single alignment.
- Both valid $\Rightarrow$ guaranteed to increase (Viterbi) likelihood.

When To Use One or the Other?

- Use Viterbi-style when can $\Rightarrow$ cheaper.
  - Optimization: need not realign every iteration.
- Intuitively, full FB may find better maxima . . .
  - But if posteriors very sharp, do almost same thing.
  - Remember example posteriors in Lab 2?
- Rule of thumb:
  - When first training "new" model, use full FB.
  - Once "locked in" to local maximum, Viterbi is fine.

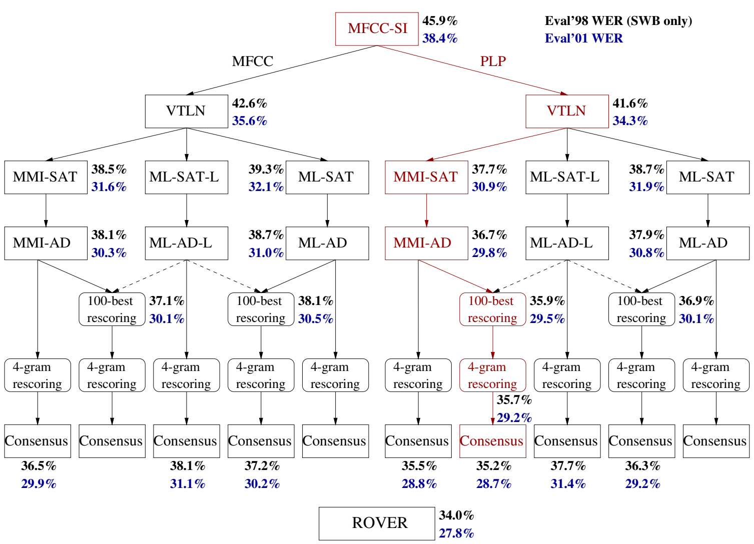
Bootstrapping One Model From Another

- Bootstrap complex model from simpler model . . .
  - Using alignment computed from simpler model.
- Point: models need not be of same form!
  - Can use WSJ model to bootstrap Switchboard model.
  - Can use triphone model to bootstrap quinphone model.
  - Can use GMM/HMM model to bootstrap DBN model.
- Requirement: same phonemes, states per phoneme.

Whew, That Was Pretty Complicated!

- The tip of the iceberg.
- Adaptation (VTLN, fMLLR, mMLLR).
- Discriminative training (LDA, MMI, MPE, fMPE).
- Model combination (cross adaptation, ROVER).

Things Can Get Pretty Hairy



How Long Does Training Take?

- It's a secret.
- Measure in terms of *real-time factor*.
  - How many hours to process one hour of speech?
- If 1,000 hours of speech, 10x real time . . .
  - How many days to train on one machine?
- Parallelization is key.
  - Data parallelization: collect FB counts on $\frac{1}{k}$th corpus.
  - Sum FB counts before parameter reestimation.

Recap

- In theory, training involves simple algorithms.
- In practice, training is insanely complicated . . .
  - For state-of-the-art systems.

Administrivia

- Clear (6); mostly clear (4).
- Pace: OK (5), slow (2).
- Muddiest: dcs trees and Gaussians (2); dcs trees and HMM's (2); criterion for constructing dcs trees (1).
- Feedback (2+ votes):
  - More info on reading project (2).
  - `http://www.ee.columbia.edu/~stanchen/fall12/e6870/readings/project_f12.html` (same password as readings).
  - Don't need to worry about this yet.

Administrivia

- Lab 2, Lab 3.
  - Not graded yet; handed back next lecture.
  - Answers: `/user1/faculty/stanchen/e6870/lab2_ans/`.
- Lab 4.
  - Postponed because material not covered yet.
  - Will announce when lab posted + new due date.
- Make-up lecture.
  - What days can you make it (same time)?
- Working on setups for non-reading projects.

Part II

Segue: Intro to LVCSR Decoding

Decoding for LVCSR

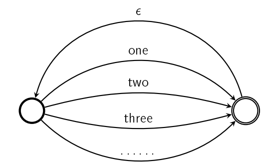
- Now know how to build models for LVCSR:
  - n-gram LM's $P(\omega)$ via counting and smoothing.
  - CD acoustic models $P_\omega(\mathbf{x})$ via complex recipes.
- This part: given test audio $\mathbf{x}$, how to compute ...
  - Most likely word sequence ω^*.

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

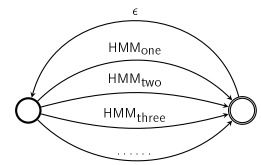
- Initially, let's ignore efficiency.
  - How to do this conceptually?

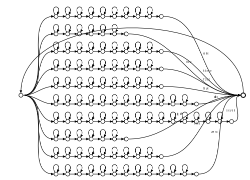
Decoding: Small Vocabulary

- Take (H)MM representing allowable word sequences/LM.



- Replace each word with corresponding HMM.





- Run Viterbi algorithm!

Can We Do Same Thing for LVCSR?

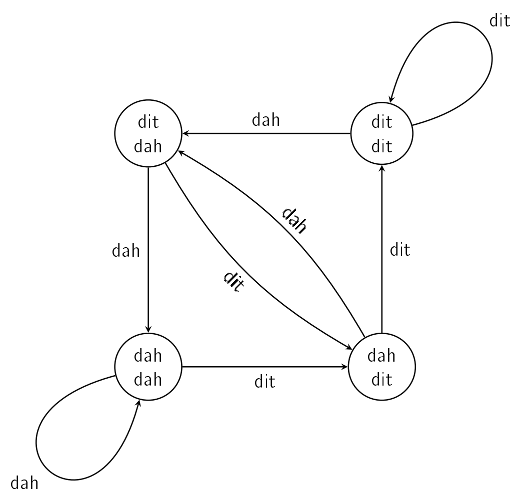
1. Can we express LM as (H)MM?
2. How to expand word HMM to full HMM?
3. Graph not too big? Not too slow to decode?

Issue 1: Is n-Gram Model an (H)MM?

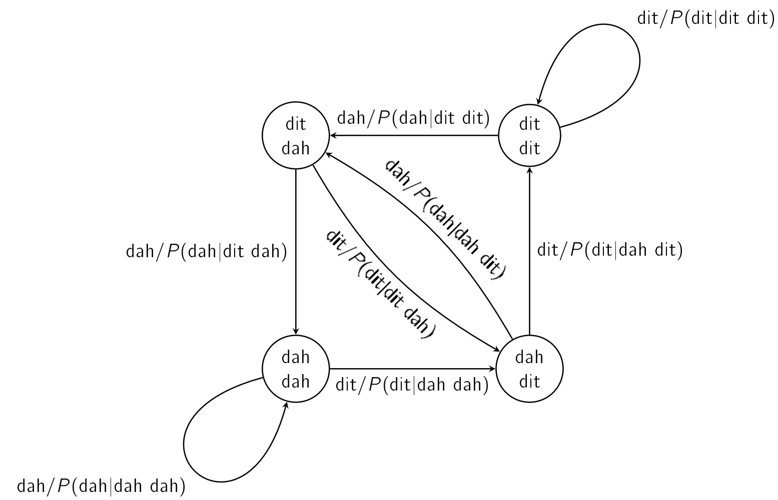
- Yup; n-gram model is Markov model of order $n-1$.
- Example: trigram model $P(w_i|w_{i-2}w_{i-1})$.
- One state for each history $w_{i-2}w_{i-1}$.
  - Arrive here iff last two words are w_{i-2}, w_{i-1}.
- Each state $w_{i-2}w_{i-1}$ has outgoing arc for every w_i ...
  - To state $w_{i-1}w_i$ with probability $P(w_i|w_{i-2}w_{i-1})$.
- For each word sequence $w_1, \ldots, w_L$ ...
  - Single path through HMM with total probability

$$P(w_1, \ldots, w_L) = \prod_i P(w_i|w_{i-2}w_{i-1})$$

Trigram LM, Morse Code, Basic Structure



Trigram LM, Morse Code, With Probabilities

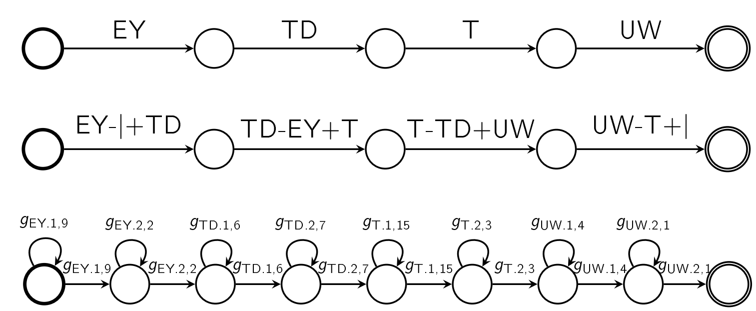


Pop Quiz

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

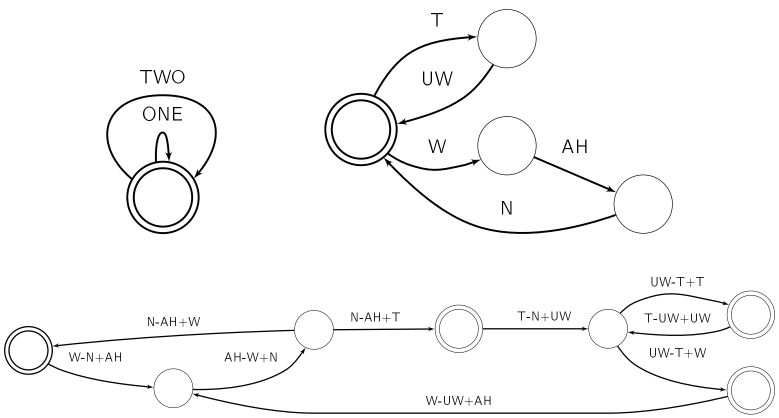
Issue 2: Graph Expansion

- Training: only single word sequence, *e.g.*, *EIGHT TWO*.



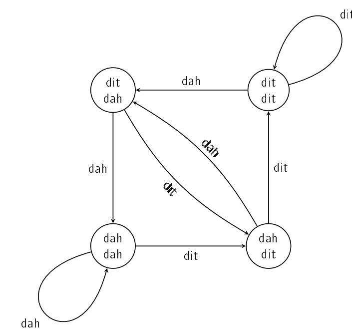
Context-Dependent Graph Expansion

- Decoding: many possible word sequences.
- CD expansion: handling branch points is tricky.
- Other issues: single-phoneme words; quinphone models.



Issue: How Big The Graph?

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



- $|V|^3$ word arcs in FSA representation.
- Say words are ~4 phones = 12 states on average.
- 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?
- PC's can do $\sim 10^{10}$ floating-point ops per second.

Recap: Small *vs.* Large Vocabulary Decoding

- In theory, can use same exact techniques.
- In practice, three big problems:
  - Context-dependent graph expansion is complicated.
  - Decoding graphs way too big.
  - Decoding way too slow.
- How can we handle this?
- Next week:
  - Finite-state machines.
  - How to make decoding efficient.

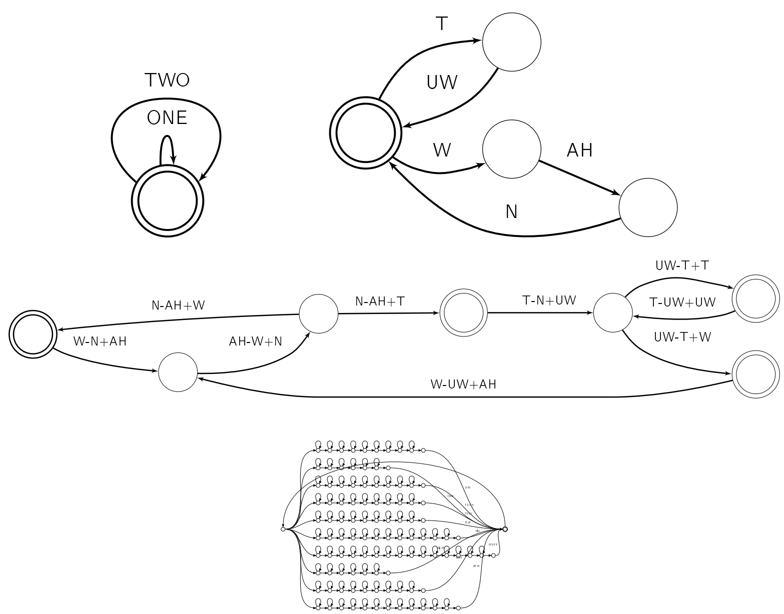
Part III

Finite-State Machines

A View of Graph Expansion

- Step 1: Take word graph as input.
  - Convert into phone graph.
- Step 2: Take phone graph as input.
  - Convert into context-dependent phone graph.
- Step 3: Take context-dependent phone graph.
  - Convert into final HMM.
- Goal: want framework for . . .
  1. Representing graphs.
  2. Transforming graphs.

A View of Graph Expansion



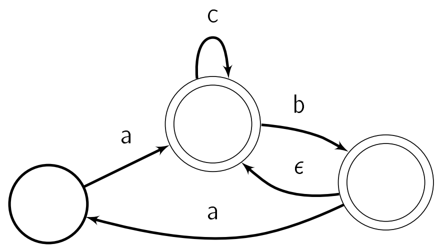
A Framework for Rewriting Graphs

- How to represent graphs?
  - HMM's ⇒ finite-state acceptors (FSA's)!
- How to represent graph transformations?
  - Finite-state transducers (FST's)!
- What operation applies transformations to graphs?
  - Composition!

Where Are We?

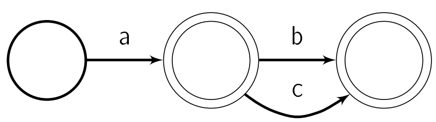
What is a Finite-State Acceptor?

- It's like an HMM, but without probabilities.
- It has states.
  - Exactly one initial state; one or more final states.
- It has arcs.
  - Each arc has a label, which may be empty (ϵ).

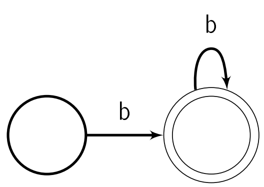


What Does an FSA *Mean*?

- The (possibly infinite) list of strings it accepts.
  - *i.e.*, strings that label path from initial to final state.
- Meaning: *a*, *ab*, *ac*.

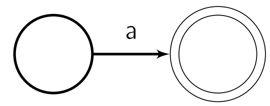


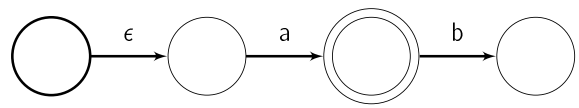
- Meaning: *b*, *bb*, *bbb*, *bbbb*, ...



Pop Quiz

- Are these equivalent?
  - *i.e.*, do they have same meaning?

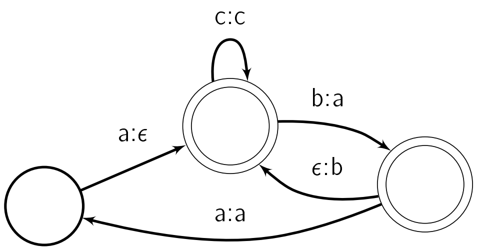




- Things that *don't* affect meaning.
  - How labels are distributed along path.
  - Invalid paths.

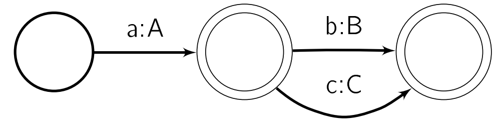
What is a Finite-State Transducer?

- It's like a finite-state acceptor, except . . .
- Each arc has two labels instead of one.
  - An *input* label (possibly empty).
  - An *output* label (possibly empty).

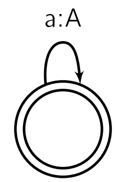


What Does an FST *Mean*?

- A (possibly infinite) list of pairs of strings . . .
  - An input string and an output string.
- Meaning: (*a*, *A*), (*ab*, *AB*), (*ac*, *AC*).

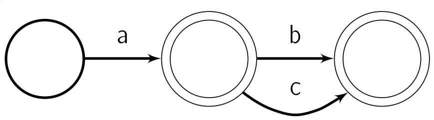


- Meaning: (ϵ, ϵ), (*b*, *a*), (*bb*, *aa*), (*bbb*, *aaa*), . . .

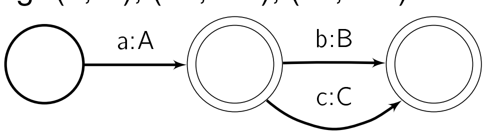


What is Composition?

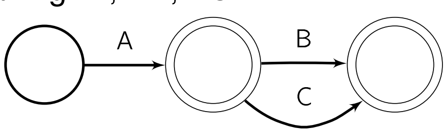
- Applying FST T to FSA A to create new FSA $A \circ T$.
  - If $\alpha \in A$ and $(\alpha, \beta) \in T$, then $\beta \in A \circ T$.
- A has meaning: *a*, *ab*, *ac*.



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



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



Recap

- *Finite-state acceptor* (FSA): one label on each arc.
- *Finite-state transducer* (FST): two labels on each arc.
- *Finite-state machine* (FSM): FSA or FST.
  - Also, *finite-state automaton*.
- FST's can be used to transform FSA's via composition.
- The point: can express each stage in graph expansion . . .
  - As applying FST via composition.

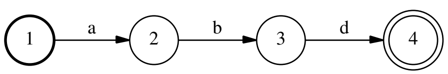
Where Are We?

The Composition Operation

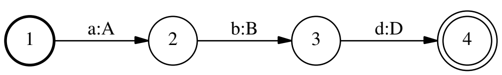
- A simple and efficient algorithm for computing . . .
  - Result of applying transducer to acceptor.
- What can composition *do*?

Rewriting Single String A Single Way

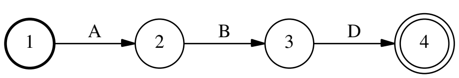
A



T

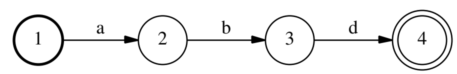


$A \circ T$

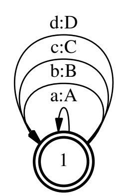


Rewriting Single String A Single Way

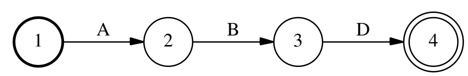
A



T



$A \circ T$



Transforming a Single String

- Let's say have string, *e.g.*,

  THE DOG
- Let's say want to apply one-to-one transformation.
  - *e.g.*, map words to their (single) baseforms.

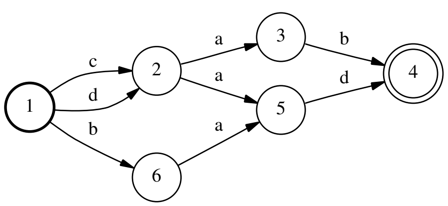
  DH AH D AO G
- This is easy, *e.g.*, use `sed` or `perl` or . . .

The Magic of FST's and Composition

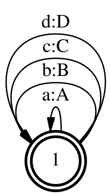
- Let's say have (possibly infinite) list of strings . . .
  - Expressed as an FSA, as this is compact.
- How to transform all strings in FSA in one go?
- How to do one-to-many or one-to-zero transformations?
- Can we express (possibly infinite) list of output strings . . .
  - As (compact) FSA?
- Fast?

Rewriting Many Strings At Once

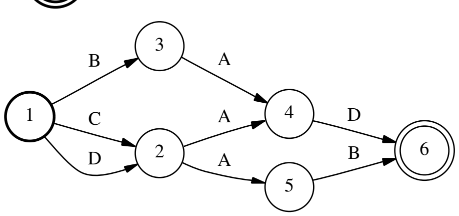
A



T

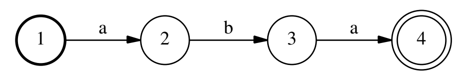


$A \circ T$

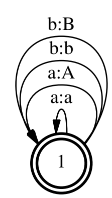


Rewriting Single String Many Ways

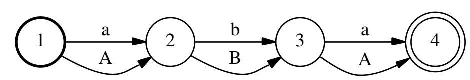
A



T

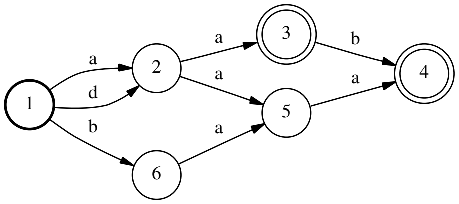


$A \circ T$



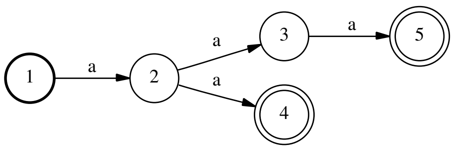
Rewriting Some Strings Zero Ways

A



T

$A \circ T$

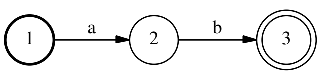


Computing Composition: The Basic Idea

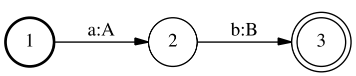
- For every state $s \in A$, $t \in T$, create state $(s, t) \in A \circ T$ ...
  - Corresponding to being in states s and t at same time.
- Make arcs in intuitive way.

Example

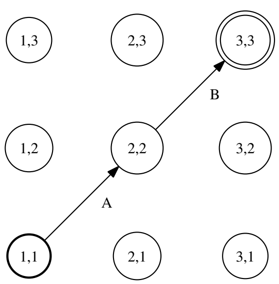
A



T



$A \circ T$



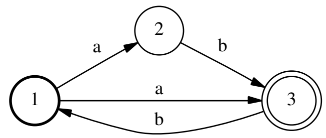
- Optimization: start from initial state, build outward.

Computing Composition: More Formally

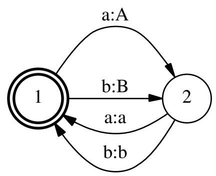
- For now, pretend no ϵ-labels.
- For every state $s \in A$, $t \in T$, create state $(s, t) \in A \circ T$.
- Create arc from (s_1, t_1) to (s_2, t_2) with label o iff ...
  - There is arc from s_1 to s_2 in A with label i and ...
  - There is arc from t_1 to t_2 in T with label $i : o$.
- (s, t) is initial iff s and t are initial; similarly for final states.
- (Remove arcs and states that are "unreachable".)
- What is time complexity?

Another Example

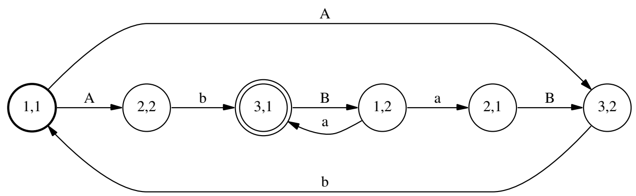
A



T



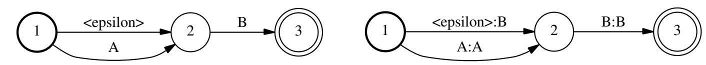
$A \circ T$



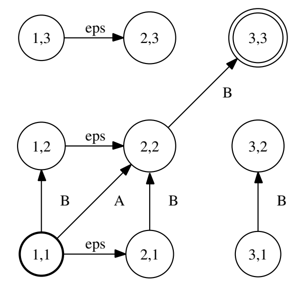
Composition and ϵ-Transitions

- 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

- FST's can express wide range of string transformations.
- Composition lets us efficiently …
  - Apply FST to all strings in FSA in one go!

FSM Toolkits

- AT&T FSM toolkit ⇒ OpenFST; lots of others.
  - Implements composition, lots of finite-state operations.
- A syntax for specifying FSA's and FST's, *e.g.*,

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