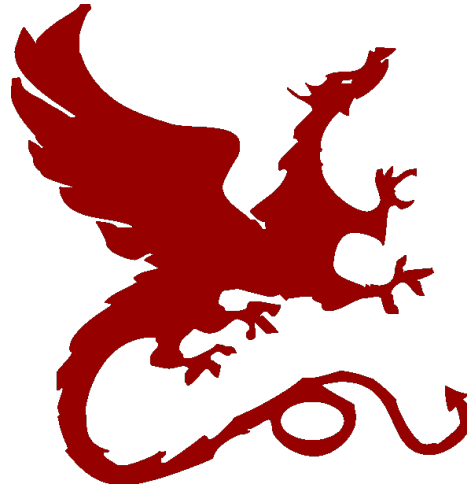


Algorithms for NLP



More Speech

Taylor Berg-Kirkpatrick – CMU

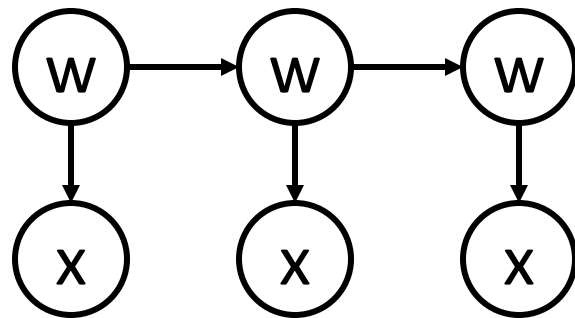
Slides: Dan Klein – UC Berkeley

State Model

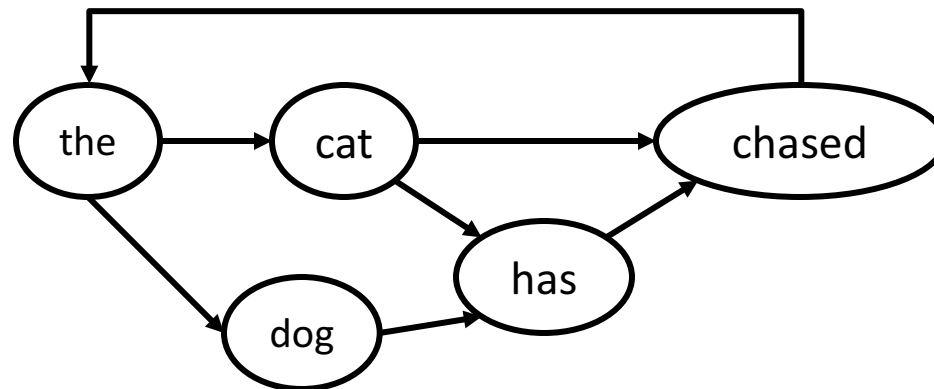


State Transition Diagrams

- Bayes Net: HMM as a Graphical Model

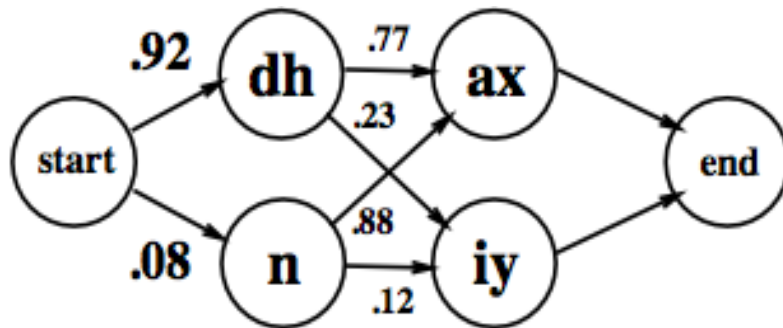


- State Transition Diagram: Markov Model as a Weighted FSA

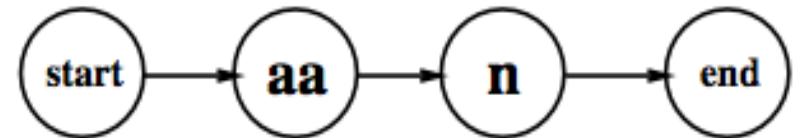




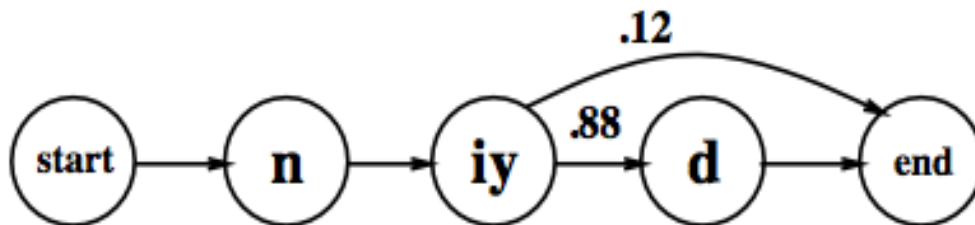
ASR Lexicon



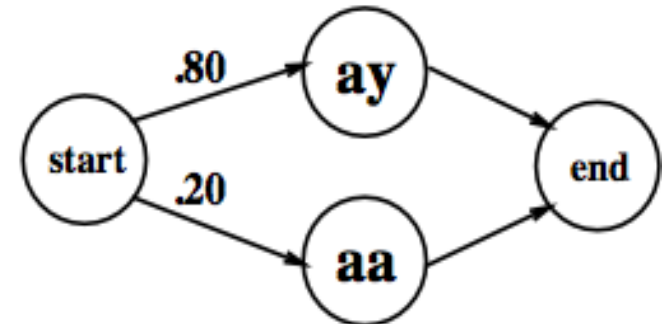
Word model for "the"



Word model for "on"



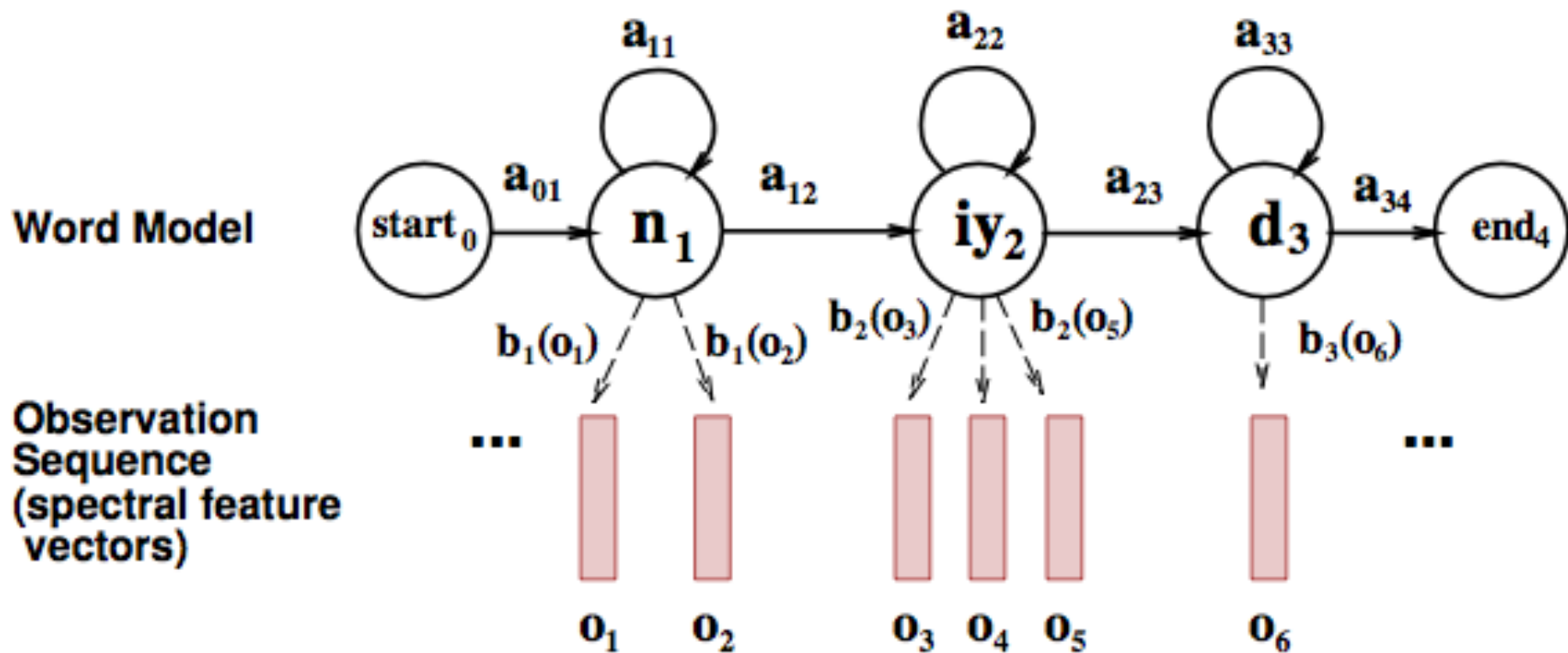
Word model for "need"



Word model for "I"

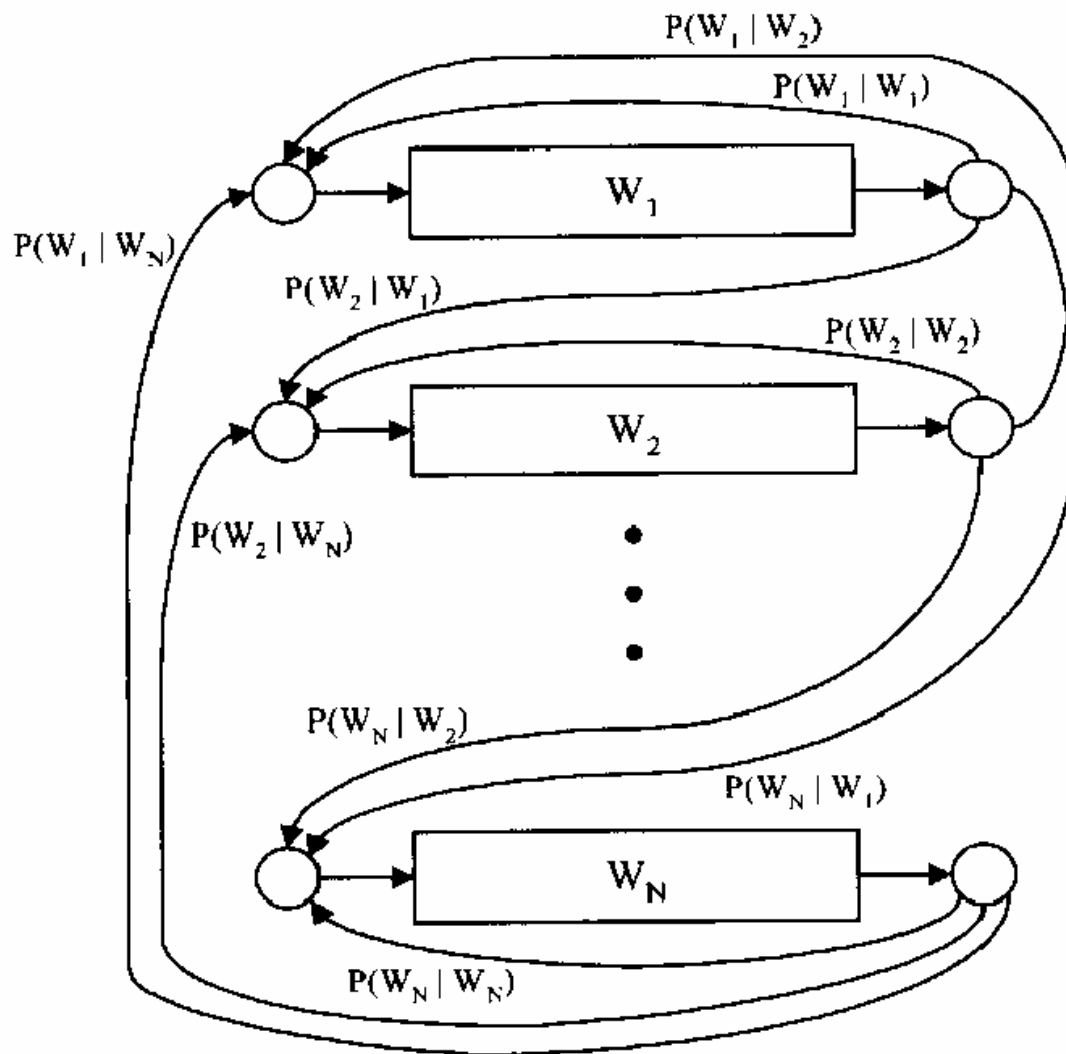


Lexical State Structure





Adding an LM





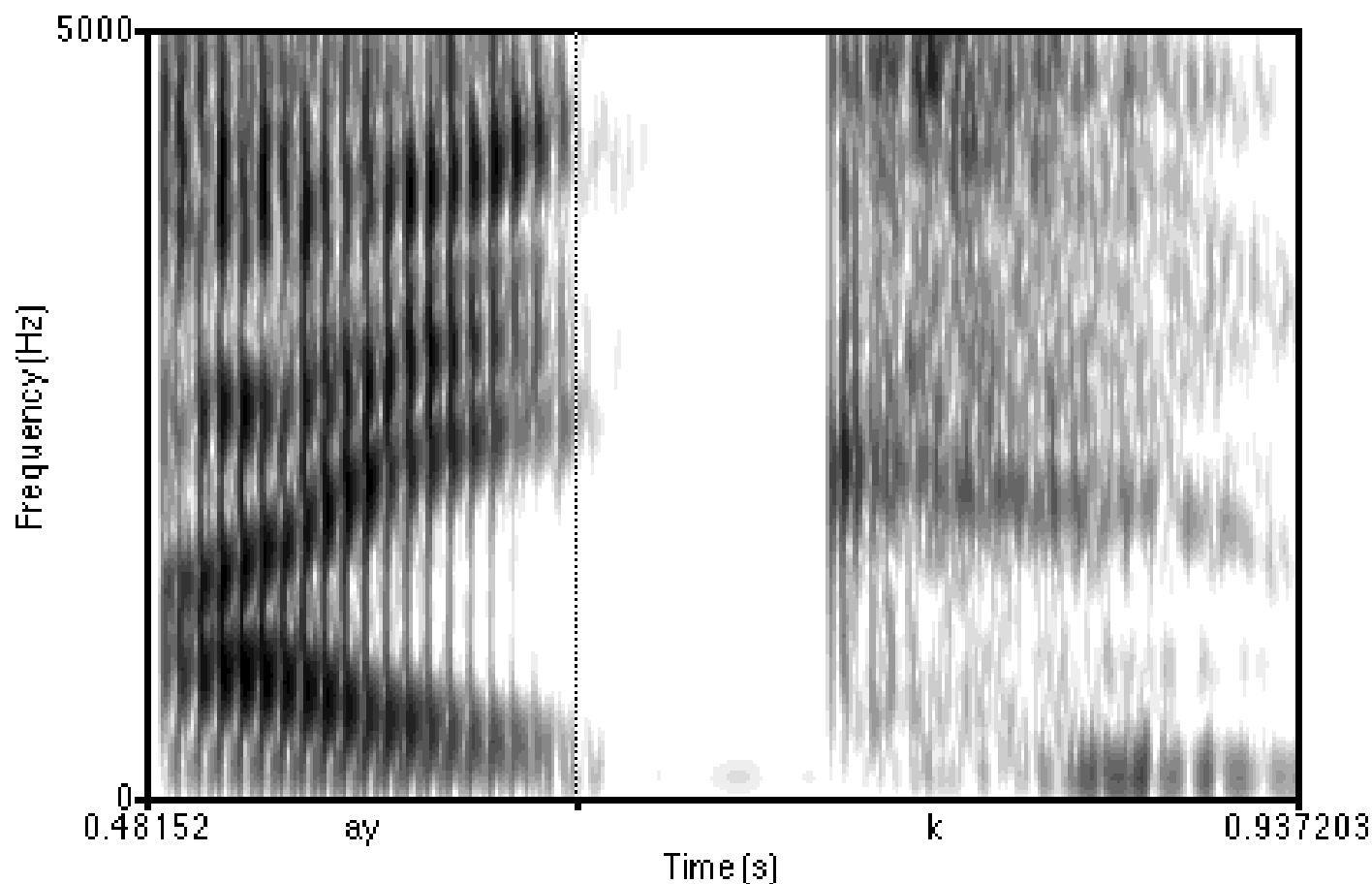
State Space

- State space must include
 - Current word ($|V|$ on order of 20K+)
 - Index within current word ($|L|$ on order of 5)
 - E.g. (lec[t]ure) (though not in orthography!)
- Acoustic probabilities only depend on phone type
 - E.g. $P(x|\text{lec}[\text{t}]ure) = P(x|t)$
- From a state sequence, can read a word sequence

State Refinement

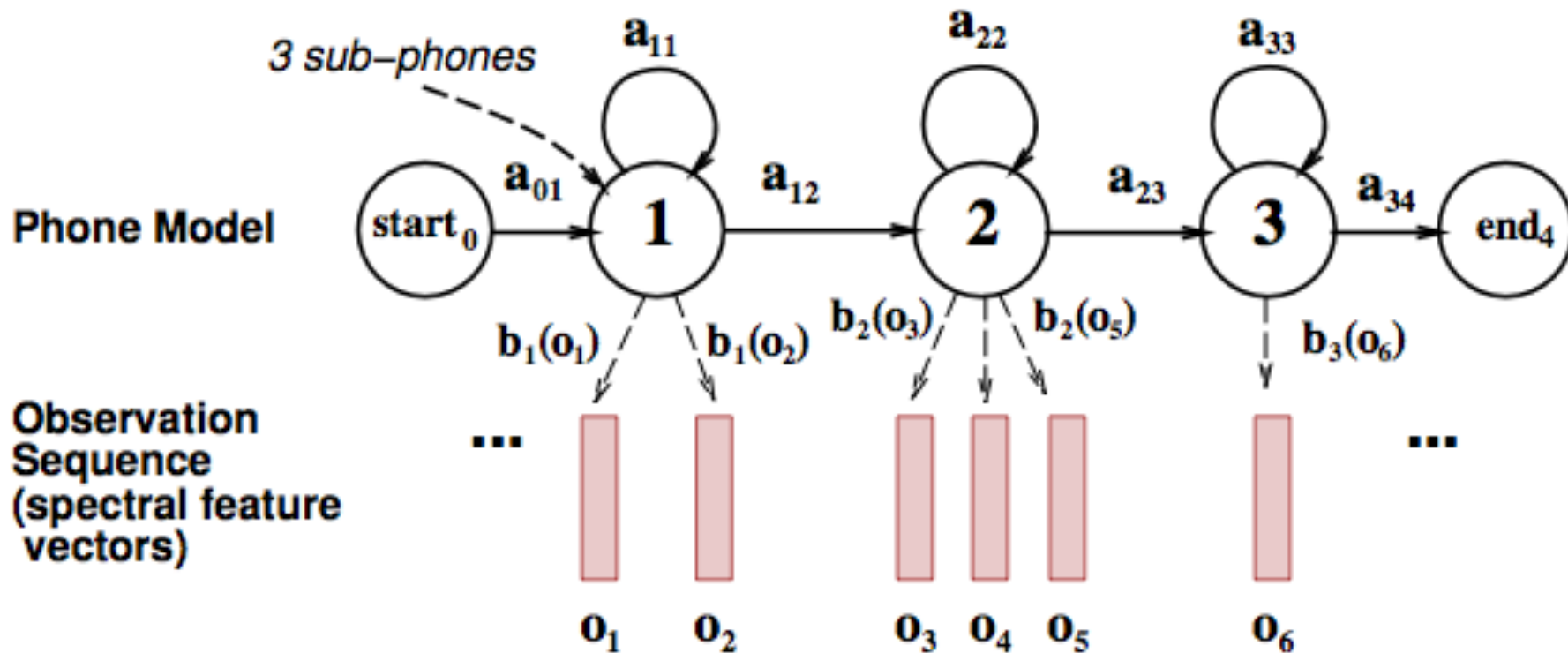


Phones Aren't Homogeneous



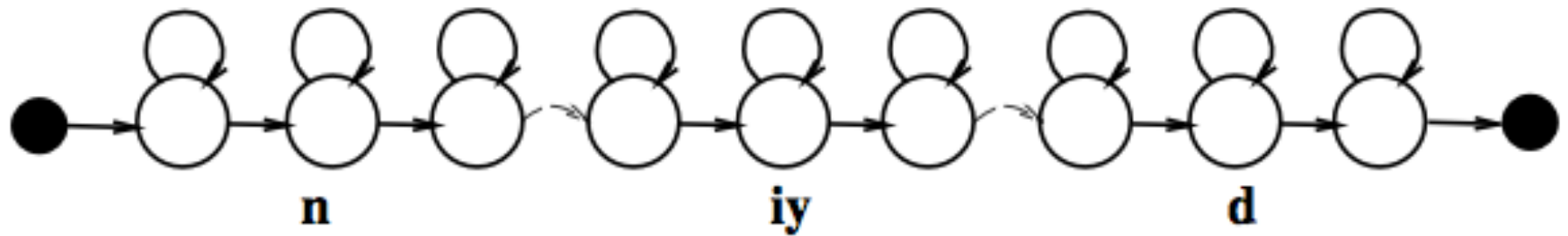


Need to Use Subphones



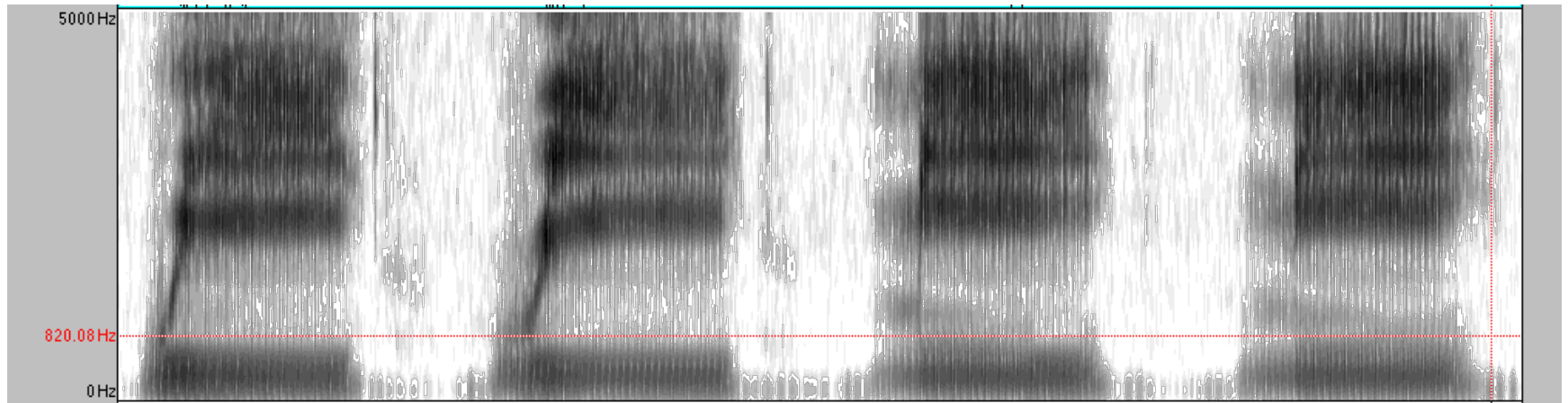


A Word with Subphones





Modeling phonetic context



w iy

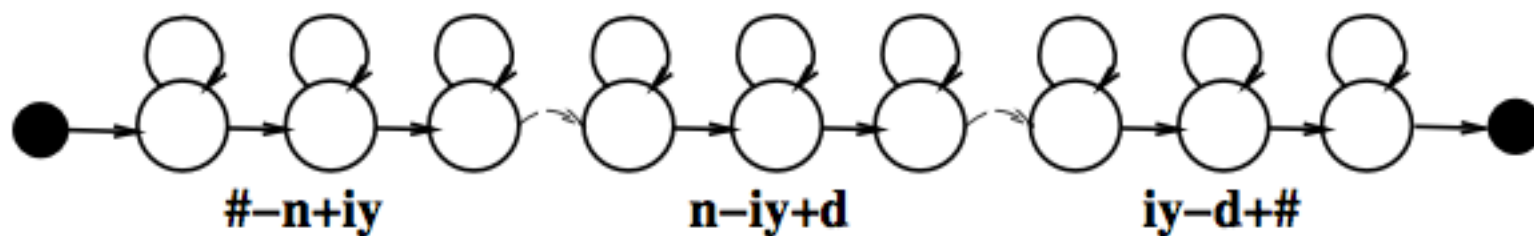
r iy

m iy

n iy



“Need” with triphone models





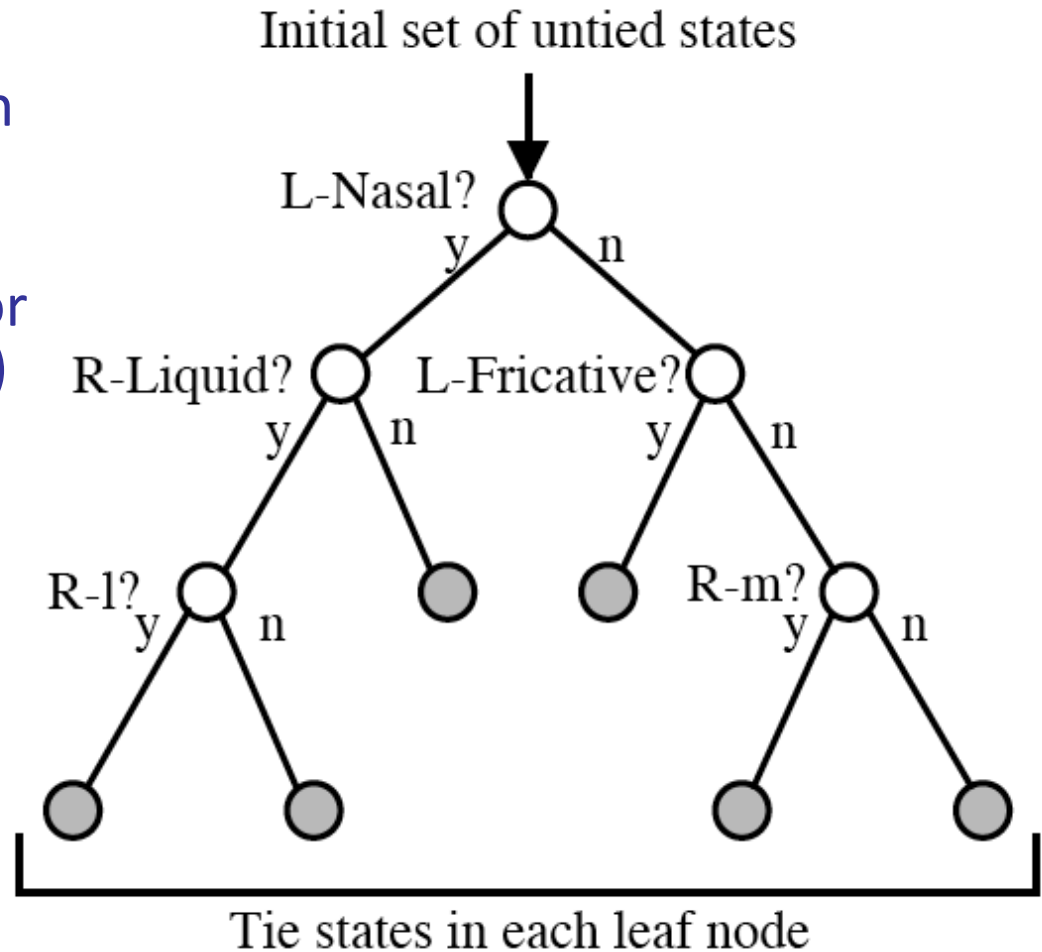
Lots of Triphones

- Possible triphones: $50 \times 50 \times 50 = 125,000$
- How many triphone types actually occur?
- 20K word WSJ Task (from Bryan Pellom)
 - Word internal models: need 14,300 triphones
 - Cross word models: need 54,400 triphones
- Need to generalize models, tie triphones



State Tying / Clustering

- [Young, Odell, Woodland 1994]
- How do we decide which triphones to cluster together?
- Use **phonetic features** (or 'broad phonetic classes')
 - Stop
 - Nasal
 - Fricative
 - Sibilant
 - Vowel
 - lateral





State Space

- State space now includes
 - Current word: $|W|$ is order 20K
 - Index in current word: $|L|$ is order 5
 - Subphone position: 3
 - E.g. (lec[t-mid]ure)
- Acoustic model depends on clustered phone context
 - But this doesn't grow the state space
- But, adding the LM context for trigram+ does
 - (after the, lec[t-mid]ure)
 - This is a real problem for decoding

Decoding



Inference Tasks



Most likely word sequence:

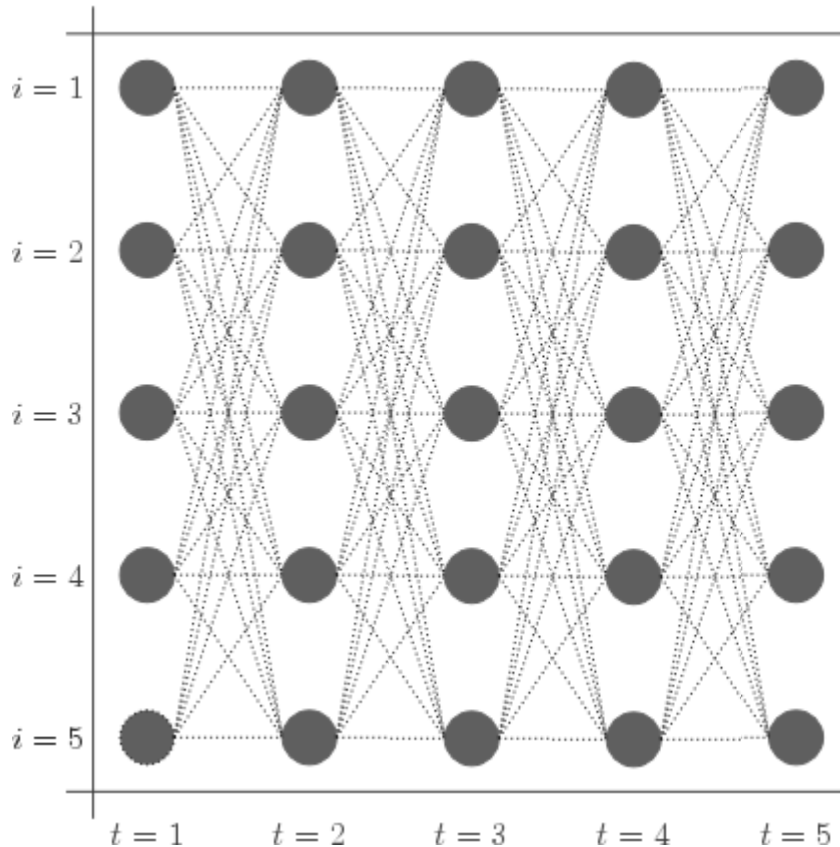
d - ae - d

Most likely state sequence:

$d_1-d_6-d_6-d_4-ae_5-ae_2-ae_3-ae_0-d_2-d_2-d_3-d_7-d_5$



Viterbi Decoding

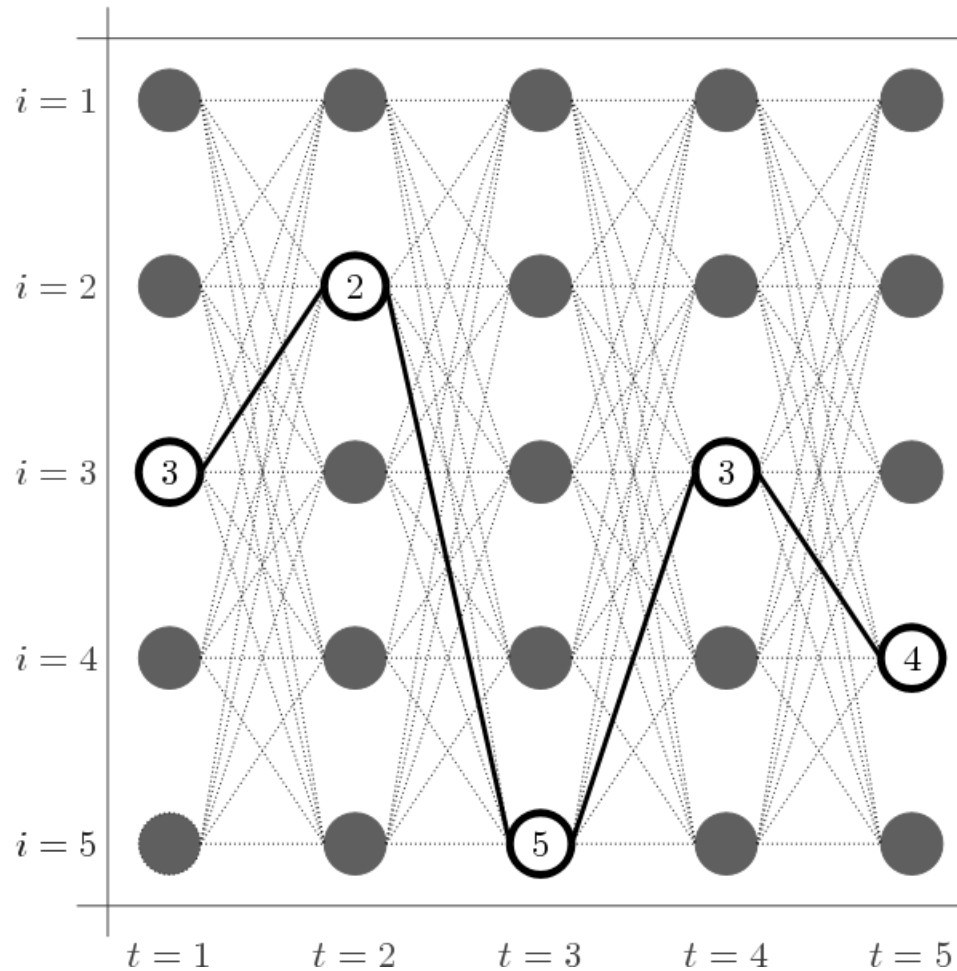


$$\phi_t(s_t, s_{t-1}) = P(x_t|s_t)P(s_t|s_{t-1})$$

$$v_t(s_t) = \max_{s_{t-1}} \phi_t(s_t, s_{t-1})v_{t-1}(s_{t-1})$$



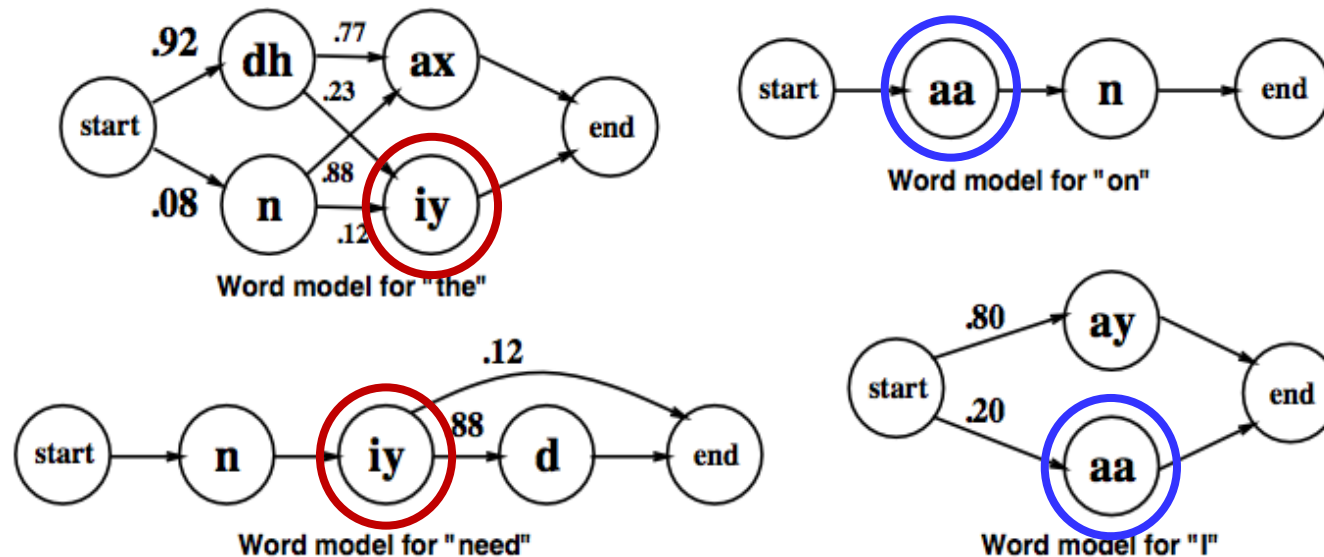
Viterbi Decoding





Emission Caching

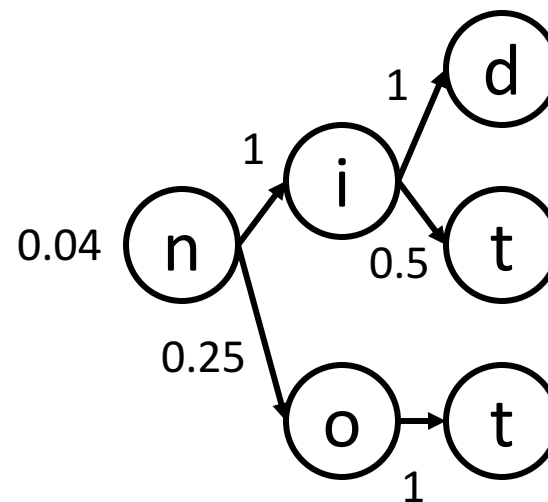
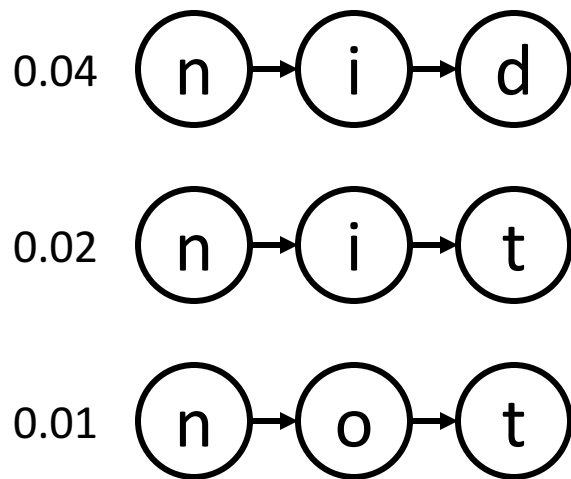
- Problem: scoring all the $P(x|s)$ values is too slow
- Idea: many states share tied emission models, so cache them





Prefix Trie Encodings

- Problem: many partial-word states are indistinguishable
- Solution: encode word production as a prefix trie (with pushed weights)

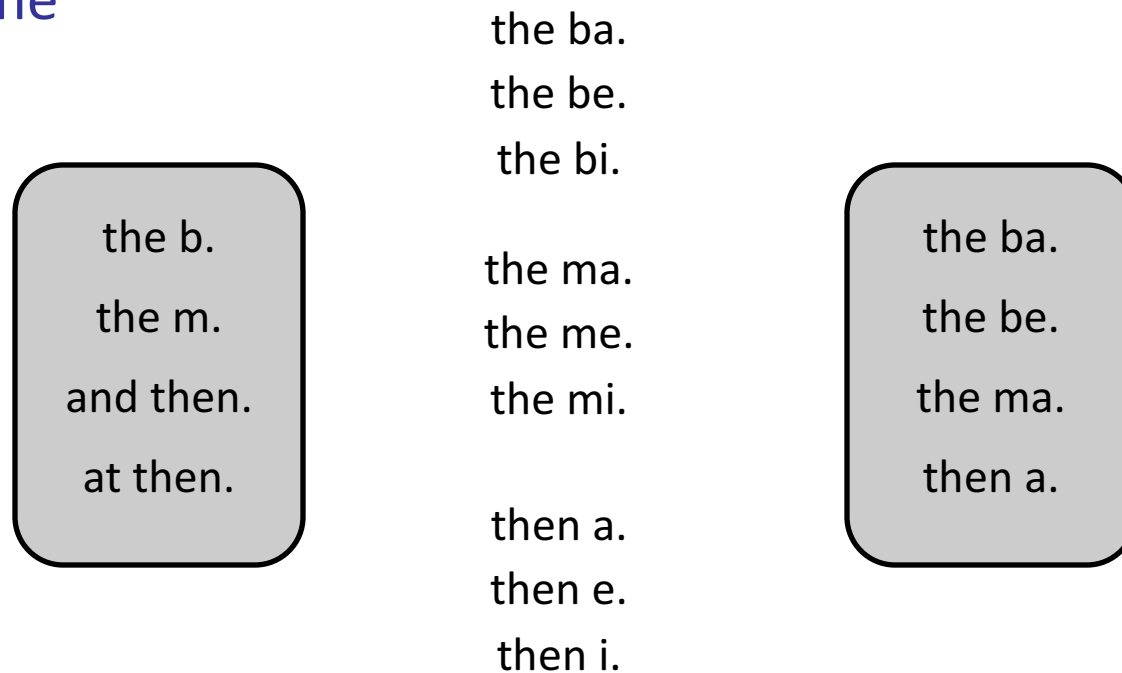


- A specific instance of minimizing weighted FSAs [Mohri, 94]



Beam Search

- Problem: trellis is too big to compute $v(s)$ vectors
- Idea: most states are terrible, keep $v(s)$ only for top states at each time

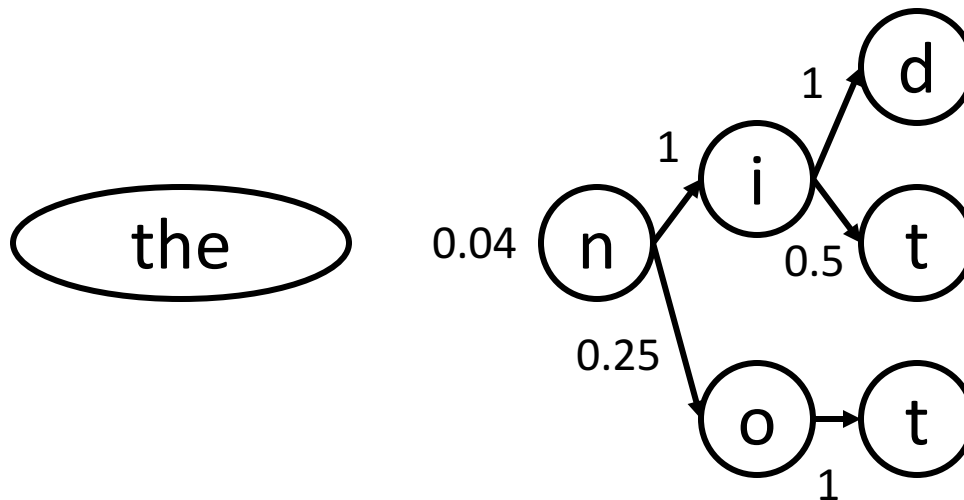


- Important: still dynamic programming; collapse equiv states



LM Factoring

- Problem: Higher-order n-grams explode the state space
- (One) Solution:
 - Factor state space into (word index, lm history)
 - Score unigram prefix costs while inside a word
 - Subtract unigram cost and add trigram cost once word is complete





LM Reweighting

- Noisy channel suggests

$$P(x|w)P(w)$$

- In practice, want to boost LM

$$P(x|w)P(w)^\alpha$$

- Also, good to have a “word bonus” to offset LM costs

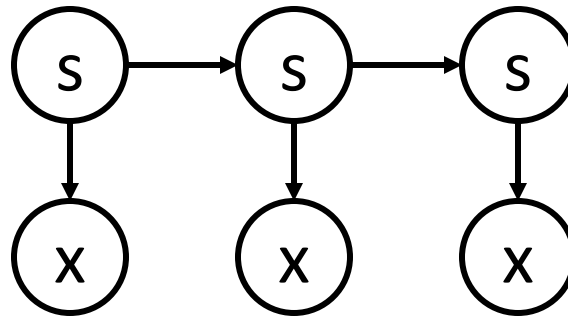
$$P(x|w)P(w)^\alpha |w|^\beta$$

- These are both consequences of broken independence assumptions in the model

Speech Training



What Needs to be Learned?

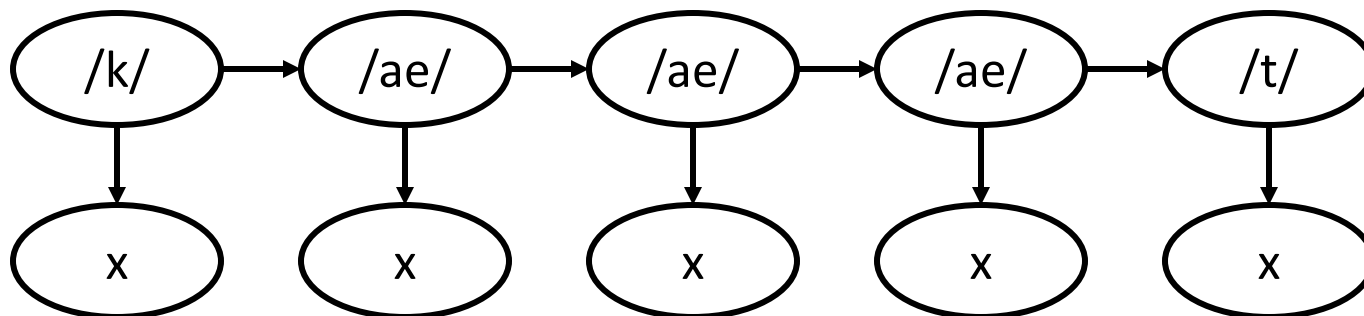


- Emissions: $P(x \mid \text{phone class})$
 - X is MFCC-valued
- Transitions: $P(\text{state} \mid \text{prev state})$
 - If between words, this is $P(\text{word} \mid \text{history})$
 - If inside words, this is $P(\text{advance} \mid \text{phone class})$
 - (Really a hierarchical model)



Estimation from Aligned Data

- What if each time step was labeled with its (context-dependent sub) phone?



- Can estimate $P(x|/ae/)$ as empirical mean and (co-)variance of x 's with label /ae/
- Problem: Don't know alignment at the frame and phone level

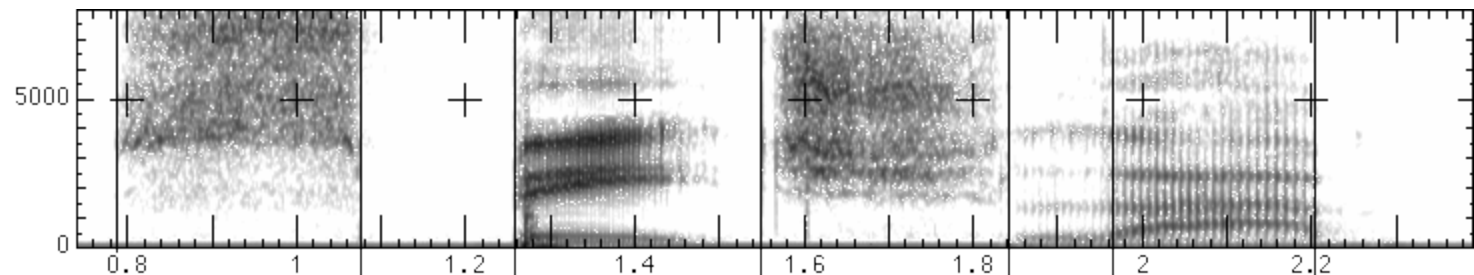


Forced Alignment

- What if the acoustic model $P(x|\text{phone})$ was known?
 - ... and also the correct sequences of words / phones
- Can predict the best alignment of frames to phones

“speech lab”

sssssssspppppeeeeeetshshshshlllllaeaeaeBBBBBB

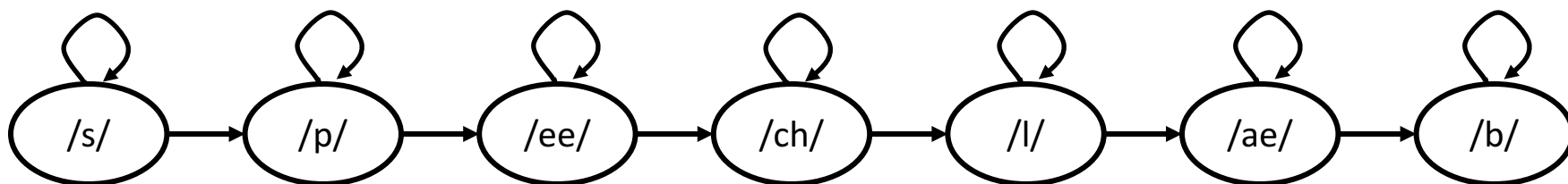


- Called “forced alignment”



Forced Alignment

- Create a new state space that forces the hidden variables to transition through phones in the (known) order



- Still have uncertainty about durations
- In this HMM, all the parameters are known
 - Transitions determined by known utterance
 - Emissions assumed to be known
 - Minor detail: self-loop probabilities
- Just run Viterbi (or approximations) to get the best alignment



EM for Alignment

- Input: acoustic sequences with word-level transcriptions
- We don't know either the emission model or the frame alignments
- Expectation Maximization (Hard EM for now)
 - Alternating optimization
 - Impute completions for unlabeled variables (here, the states at each time step)
 - Re-estimate model parameters (here, Gaussian means, variances, mixture ids)
 - Repeat
 - One of the earliest uses of EM!

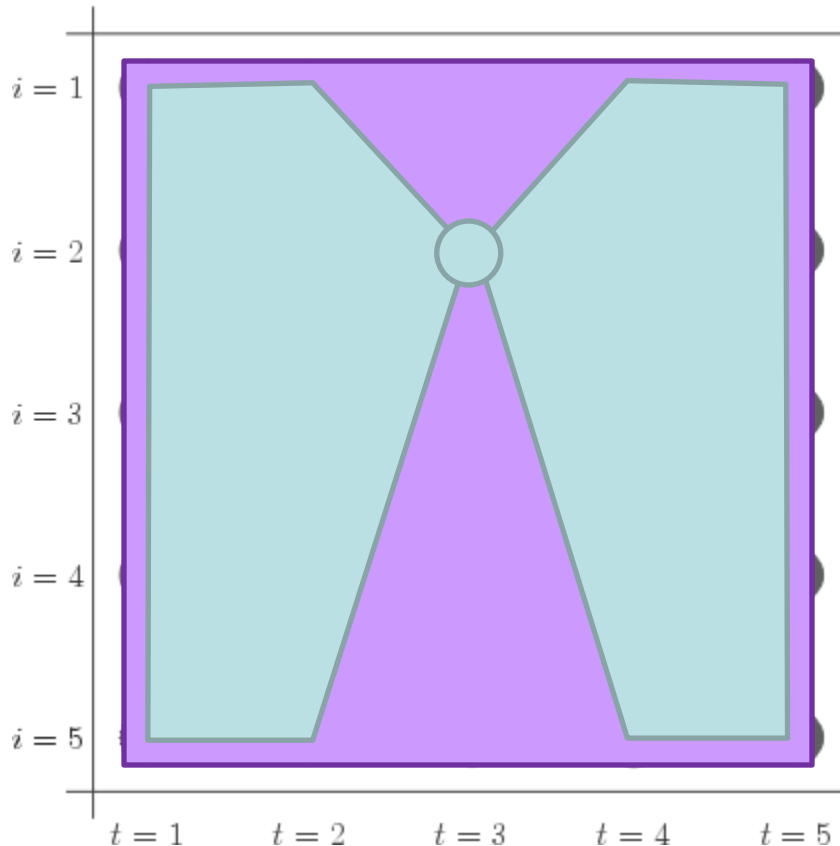


Soft EM

- Hard EM uses the best single completion
 - Here, single best alignment
 - Not always representative
 - Certainly bad when your parameters are initialized and the alignments are all tied
 - Uses the count of various configurations (e.g. how many tokens of /ae/ have self-loops)
- What we'd really like is to know the fraction of paths that include a given completion
 - E.g. 0.32 of the paths align this frame to /p/, 0.21 align it to /ee/, etc.
 - Formally want to know the expected count of configurations
 - Key quantity: $P(s_t \mid x)$



Computing Marginals

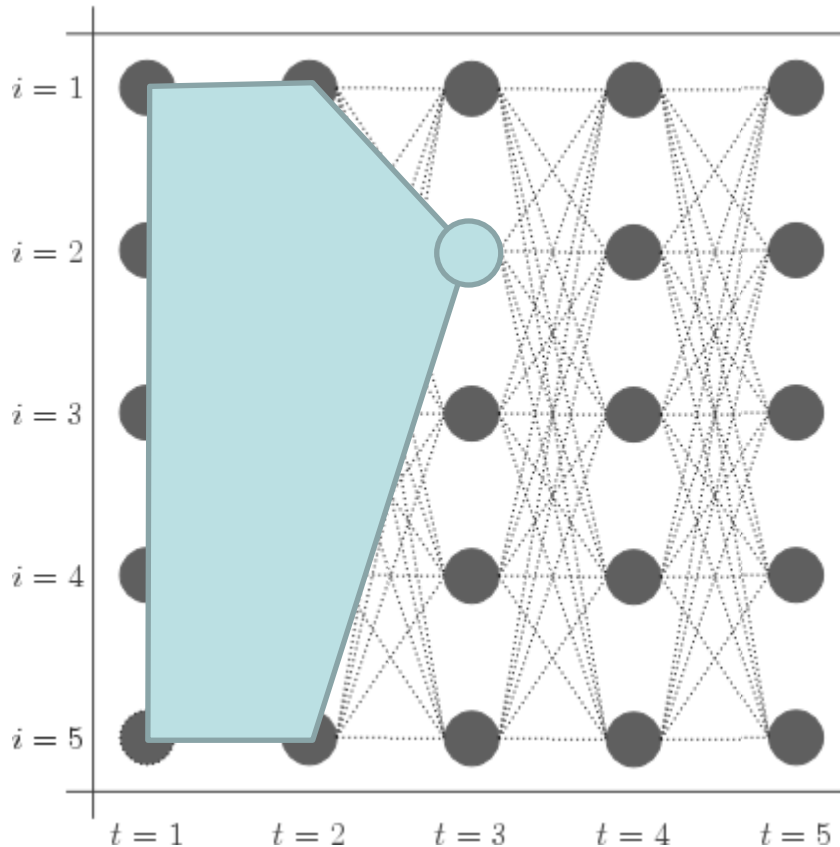


$$P(s_t|x) = \frac{P(s_t, x)}{P(x)}$$

= sum of all paths through s at t
sum of all paths



Forward Scores

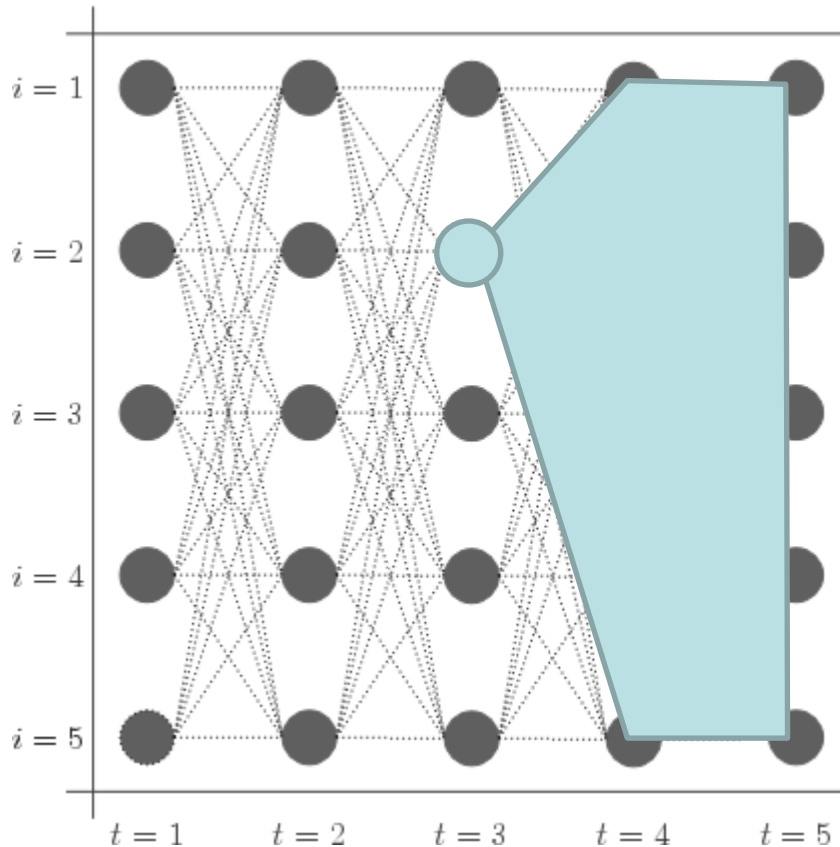


$$v_t(s_t) = \max_{s_{t-1}} v_{t-1}(s_{t-1}) \phi_t(s_{t-1}, s_t)$$

$$\alpha_t(s_t) = \sum_{s_{t-1}} \alpha_{t-1}(s_{t-1}) \phi_t(s_{t-1}, s_t)$$



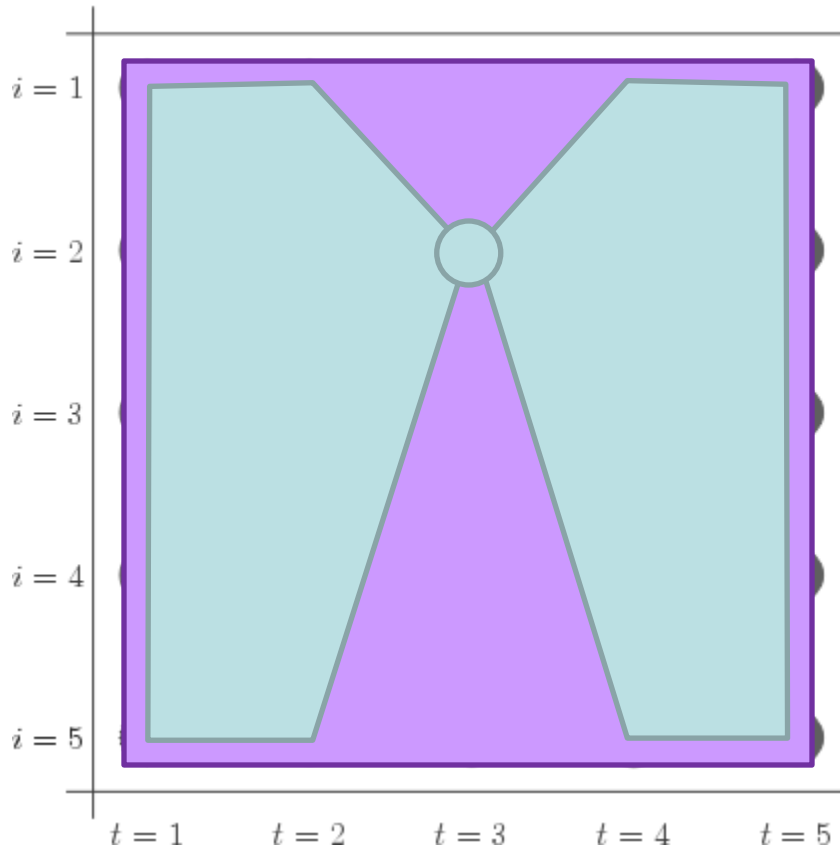
Backward Scores



$$\beta_t(s_t) = \sum_{s_{t+1}} \beta_{t+1}(s_{t+1}) \phi_t(s_t, s_{t+1})$$



Total Scores



$$P(s_t, x) = \alpha_t(s_t)\beta_t(s_t)$$

$$P(x) = \sum_{s_t} \alpha_t(s_t)\beta_t(s_t)$$

$$= \alpha_T(\text{stop})$$

$$= \beta_0(\text{start})$$



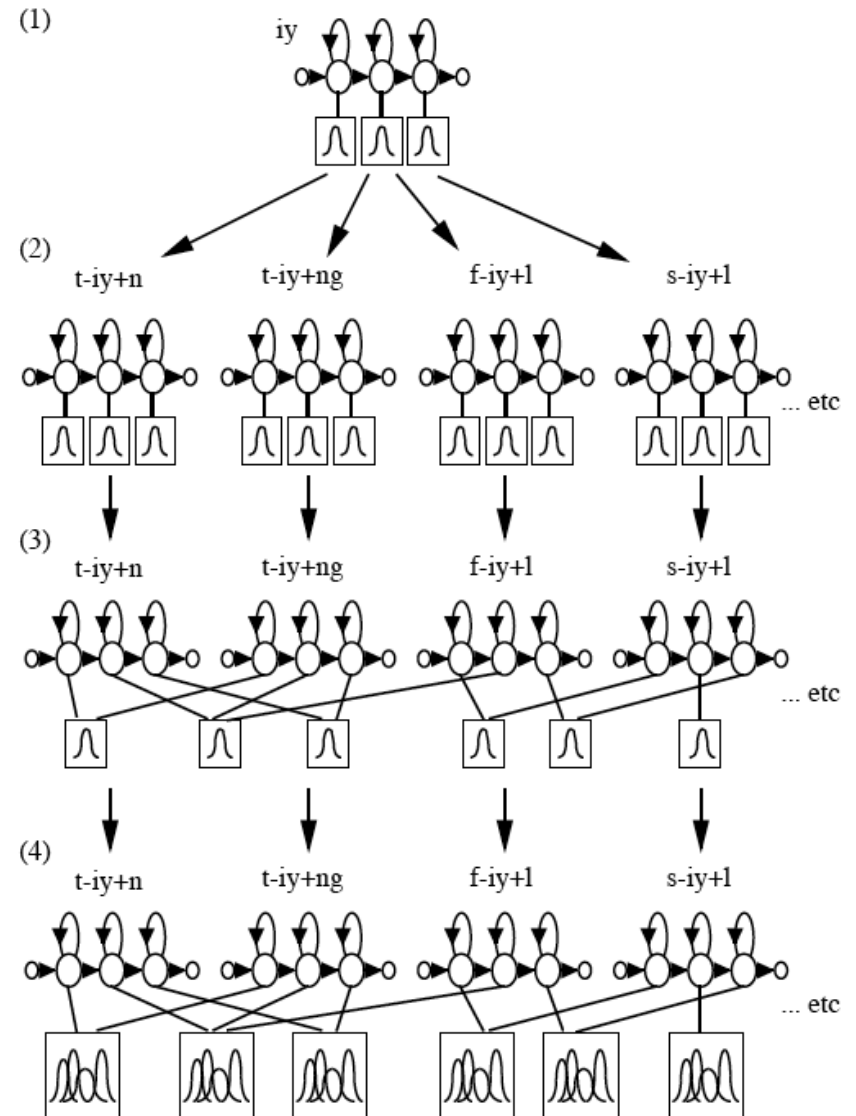
Fractional Counts

- Computing fractional (expected) counts
 - Compute forward / backward probabilities
 - For each position, compute marginal posteriors
 - Accumulate expectations
 - Re-estimate parameters (e.g. means, variances, self-loop probabilities) from ratios of these expected counts



Staged Training and State Tying

- Creating CD phones:
 - Start with monophone, do EM training
 - Clone Gaussians into triphones
 - Build decision tree and cluster Gaussians
 - Clone and train mixtures (GMMs)
- General idea:
 - Introduce complexity gradually
 - Interleave constraint with flexibility

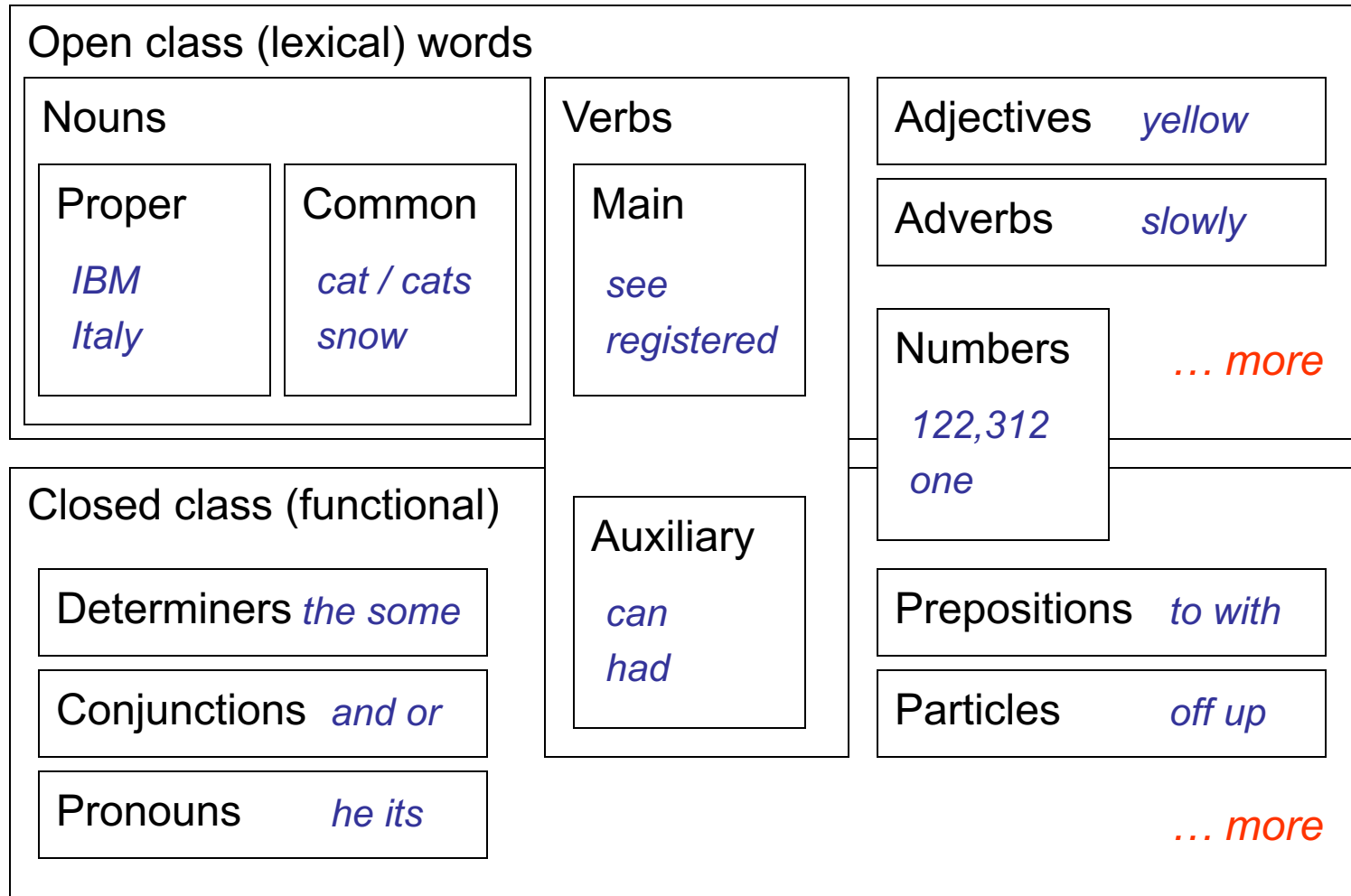


Parts of Speech



Parts-of-Speech (English)

- One basic kind of linguistic structure: syntactic word classes



| | | |
|--------------|--|---|
| CC | conjunction, coordinating | and both but either or |
| CD | numeral, cardinal | mid-1890 nine-thirty 0.5 one |
| DT | determiner | a all an every no that the |
| EX | existential there | there |
| FW | foreign word | gemeinschaft hund ich jeux |
| IN | preposition or conjunction, subordinating | among whether out on by if |
| JJ | adjective or numeral, ordinal | third ill-mannered regrettable |
| JJR | adjective, comparative | braver cheaper taller |
| JJS | adjective, superlative | bravest cheapest tallest |
| MD | modal auxiliary | can may might will would |
| NN | noun, common, singular or mass | cabbage thermostat investment subhumanity |
| NNP | noun, proper, singular | Motown Cougar Yvette Liverpool |
| NNPS | noun, proper, plural | Americans Materials States |
| NNS | noun, common, plural | undergraduates bric-a-brac averages |
| POS | genitive marker | ' 's |
| PRP | pronoun, personal | hers himself it we them |
| PRP\$ | pronoun, possessive | her his mine my our ours their thy your |
| RB | adverb | occasionally maddeningly adventurously |
| RBR | adverb, comparative | further gloomier heavier less-perfectly |
| RBS | adverb, superlative | best biggest nearest worst |
| RP | particle | aboard away back by on open through |
| TO | "to" as preposition or infinitive marker | to |
| UH | interjection | huh howdy uh whammo shucks heck |
| VB | verb, base form | ask bring fire see take |
| VBD | verb, past tense | pleaded swiped registered saw |
| VBG | verb, present participle or gerund | stirring focusing approaching erasing |
| VBN | verb, past participle | dilapidated imitated reunified unsettled |
| VBP | verb, present tense, not 3rd person singular | twist appear comprise mold postpone |
| VBZ | verb, present tense, 3rd person singular | bases reconstructs marks uses |
| WDT | WH-determiner | that what whatever which whichever |
| WP | WH-pronoun | that what whatever which who whom |
| WP\$ | WH-pronoun, possessive | whose |
| WRB | Wh-adverb | however whenever where why |



Part-of-Speech Ambiguity

- Words can have multiple parts of speech

| | | | | | | |
|-----|-----|-----|-----|----|----|--|
| VBD | | VB | | | | |
| VBN | VBZ | VBP | VBZ | | | |
| NNP | NNS | NN | NNS | CD | NN | |

Fed raises interest rates 0.5 percent

Mrs./NNP Shaefer/NNP never/RB got/VBD **around/RP** to/TO joining/VBG

All/DT we/PRP gotta/VBN do/VB is/VBZ go/VB **around/IN** the/DT corner/NN

Chateau/NNP Petrus/NNP costs/VBZ **around/RB** 250/CD

- Two basic sources of constraint:
 - Grammatical environment
 - Identity of the current word
- Many more possible features:
 - Suffixes, capitalization, name databases (gazetteers), etc...



Why POS Tagging?

- Useful in and of itself (more than you'd think)
 - Text-to-speech: record, lead
 - Lemmatization: saw[v] → see, saw[n] → saw
 - Quick-and-dirty NP-chunk detection: `grep {JJ | NN}* {NN | NNS}`
- Useful as a pre-processing step for parsing
 - Less tag ambiguity means fewer parses
 - However, some tag choices are better decided by parsers

DT NNP NN VBD VBN **IN** RP NN NNS
The Georgia branch had taken **on** loan commitments ...

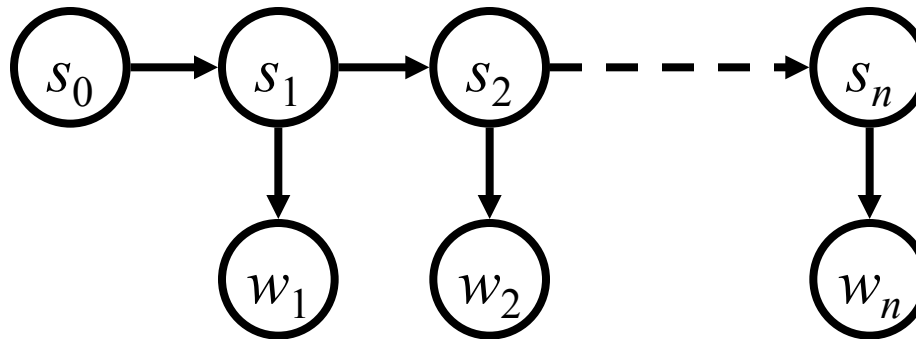
DT NN IN NN **VDN** VBD NNS VBD
The average of interbank **offered** rates plummeted ...

Part-of-Speech Tagging



Classic Solution: HMMs

- We want a model of sequences s and observations w



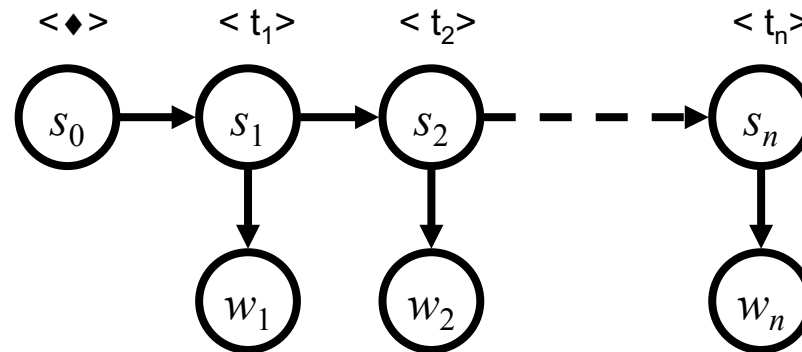
$$P(s, w) = \prod_i P(s_i | s_{i-1}) P(w_i | s_i)$$

- Assumptions:
 - States are tag n-grams
 - Usually a dedicated start and end state / word
 - Tag/state sequence is generated by a markov model
 - Words are chosen independently, conditioned only on the tag/state
 - These are totally broken assumptions: why?

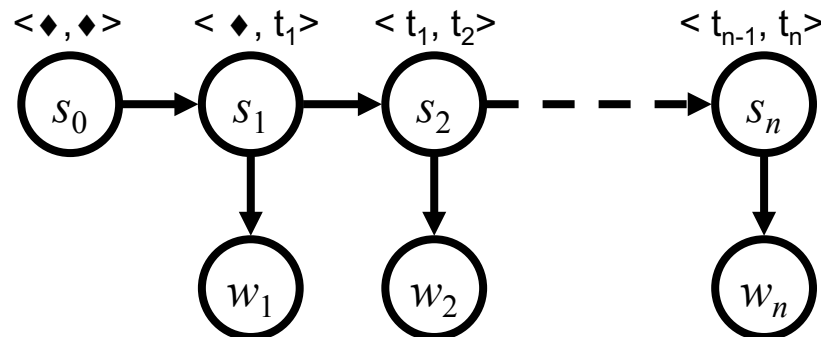


States

- States encode what is relevant about the past
- Transitions $P(s|s')$ encode well-formed tag sequences
 - In a bigram tagger, states = tags



- In a trigram tagger, states = tag pairs





Estimating Transitions

- Use standard smoothing methods to estimate transitions:

$$P(t_i | t_{i-1}, t_{i-2}) = \lambda_2 \hat{P}(t_i | t_{i-1}, t_{i-2}) + \lambda_1 \hat{P}(t_i | t_{i-1}) + (1 - \lambda_1 - \lambda_2) \hat{P}(t_i)$$

- Can get a lot fancier (e.g. KN smoothing) or use higher orders, but in this case it doesn't buy much
- One option: encode more into the state, e.g. whether the previous word was capitalized (Brants 00)
- BIG IDEA: The basic approach of state-splitting / refinement turns out to be very important in a range of tasks



Estimating Emissions

$$P(s, w) = \prod_i P(s_i | s_{i-1}) P(w_i | s_i)$$

- Emissions are trickier:

- Words we've never seen before
- Words which occur with tags we've never seen them with
- One option: break out the fancy smoothing (e.g. KN, Good-Turing)
- Issue: unknown words aren't black boxes:

343,127.23 11-year Minteria reintroducibly

- Basic solution: unknown words classes (affixes or shapes)

D⁺, D⁺.D⁺ D⁺-x⁺ Xx⁺ x⁺-"ly"

- Common approach: Estimate P(t|w) and invert
- [Brants 00] used a suffix trie as its (inverted) emission model



Disambiguation (Inference)

- Problem: find the most likely (Viterbi) sequence under the model

$$t^* = \arg \max_t P(t|w)$$

- Given model parameters, we can score any tag sequence

| | | | | | | | |
|-------|---------|-----------|----------|----------|----------|---------|--------|
| <◆,◆> | <◆,NNP> | <NNP,VBZ> | <VBZ,NN> | <NN,NNS> | <NNS,CD> | <CD,NN> | <STOP> |
| | NNP | VBZ | NN | NNS | CD | NN | . |
| | Fed | raises | interest | rates | 0.5 | percent | . |

$P(\text{NNP}|\text{<◆,◆>}) P(\text{Fed}|\text{NNP}) P(\text{VBZ}|\text{<NNP,◆>}) P(\text{raises}|\text{VBZ}) P(\text{NN}|\text{VBZ,NNP})\dots$

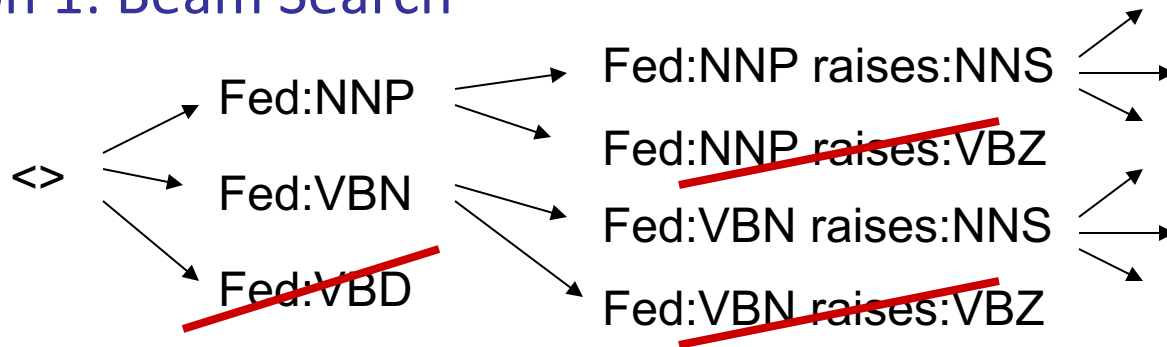
- In principle, we're done – list all possible tag sequences, score each one, pick the best one (the Viterbi state sequence)

| | | |
|----------------------|---|------------|
| NNP VBZ NN NNS CD NN | ➡ | logP = -23 |
| NNP NNS NN NNS CD NN | ➡ | logP = -29 |
| NNP VBZ VB NNS CD NN | ➡ | logP = -27 |



Finding the Best Trajectory

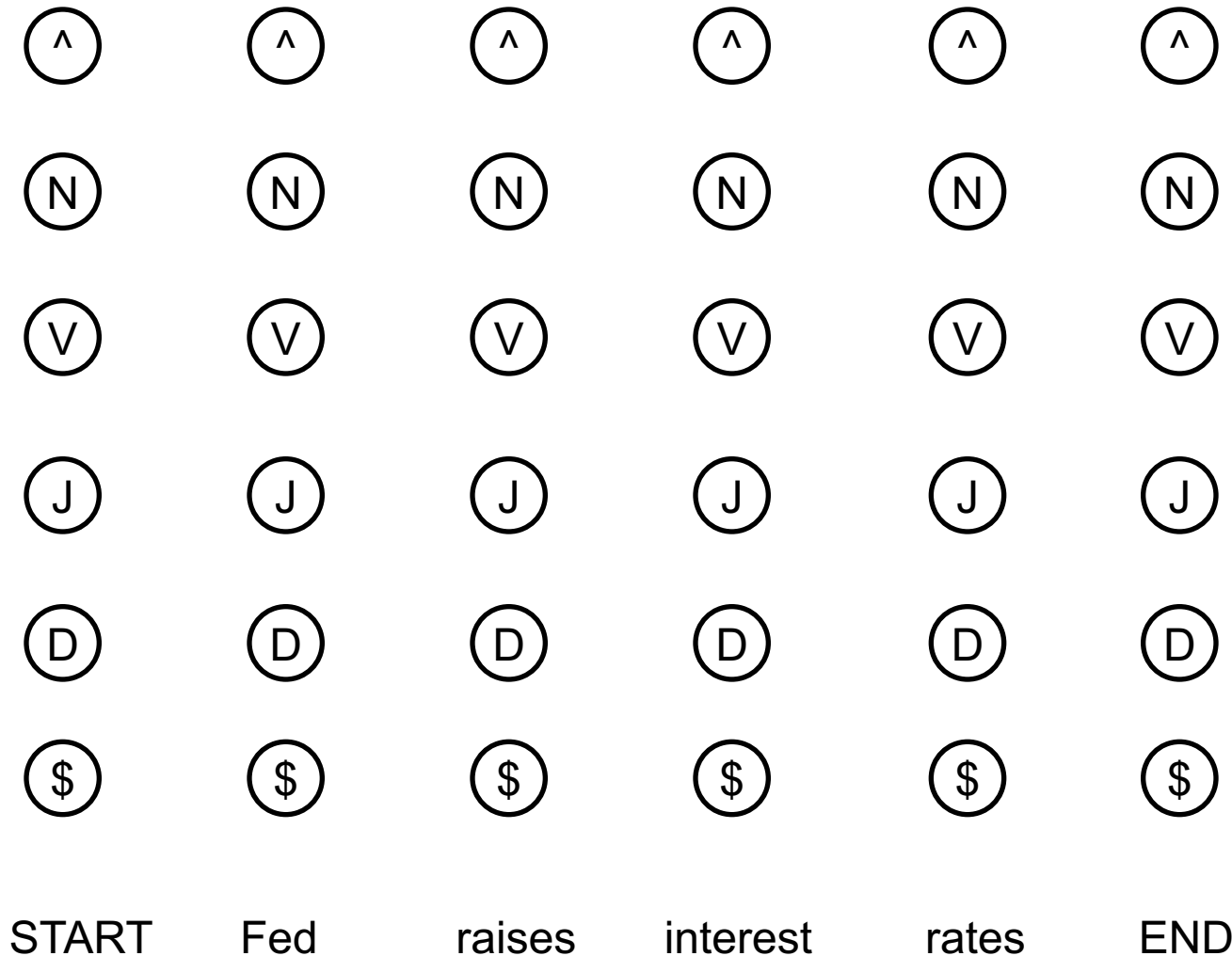
- Too many trajectories (state sequences) to list
- Option 1: Beam Search



- A beam is a set of partial hypotheses
- Start with just the single empty trajectory
- At each derivation step:
 - Consider all continuations of previous hypotheses
 - Discard most, keep top k, or those within a factor of the best
- Beam search works ok in practice
 - ... but sometimes you want the optimal answer
 - ... and you need optimal answers to validate your beam search
 - ... and there's usually a better option than naïve beams

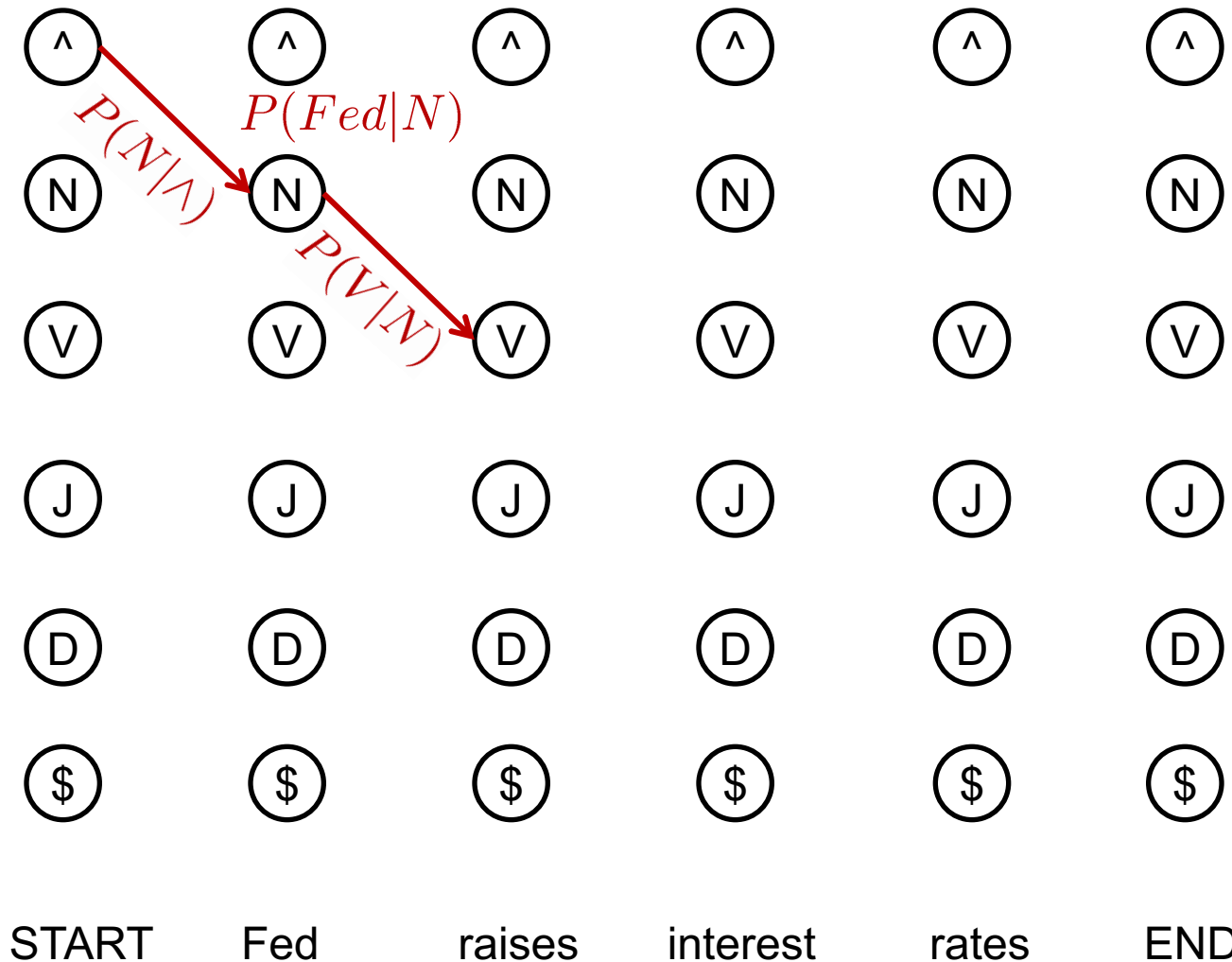


The State Lattice / Trellis





The State Lattice / Trellis





The Viterbi Algorithm

- Dynamic program for computing

$$\delta_i(s) = \max_{s_0 \dots s_{i-1}} P(s_0 \dots s_{i-1} s, w_1 \dots w_{i-1})$$

- The score of a best path up to position i ending in state s

$$\delta_0(s) = \begin{cases} 1 & \text{if } s = \langle \bullet, \bullet \rangle \\ 0 & \text{otherwise} \end{cases}$$

$$\delta_i(s) = \max_{s'} P(s | s') P(w | s') \delta_{i-1}(s')$$

- Also can store a backtrace (but no one does)

$$\psi_i(s) = \arg \max_{s'} P(s | s') P(w | s') \delta_{i-1}(s')$$

- Memoized solution
- Iterative solution



So How Well Does It Work?

- Choose the most common tag

- 90.3% with a bad unknown word model
- 93.7% with a good one

- TnT (Brants, 2000):

- A carefully smoothed trigram tagger
- Suffix trees for emissions
- 96.7% on WSJ text (SOTA is 97+%)

- Noise in the data

- Many errors in the training and test corpora

DT NN IN NN VBD NNS VBD
The average of interbank offered rates plummeted ...

- Probably about 2% guaranteed error from noise (on this data)

JJ JJ NN
chief executive officer

NN JJ NN
chief executive officer

JJ NN NN
chief executive officer

NN NN NN
chief executive officer



Overview: Accuracies

- Roadmap of (known / unknown) accuracies:

- Most freq tag: ~90% / ~50%

- Trigram HMM: ~95% / ~55%

- TnT (HMM++): 96.2% / 86.0%

- Maxent $P(t|w)$: 93.7% / 82.6%

- MEMM tagger: 96.9% / 86.9%

- State-of-the-art: 97+% / 89+%

- Upper bound: ~98%

Most errors
on unknown
words



Common Errors

- Common errors [from Toutanova & Manning 00]

| | JJ | NN | NNP | NNPS | RB | RP | IN | VB | VBD | VCN | VBP | Total |
|-------|-----|-----|-----|------|-----|-----|-----|-----|-----|-----|-----|-------|
| JJ | 0 | 177 | 56 | 0 | 61 | 2 | 5 | 10 | 15 | 108 | 0 | 488 |
| NN | 244 | 0 | 103 | 0 | 12 | 1 | 1 | 29 | 5 | 6 | 19 | 525 |
| NNP | 107 | 106 | 0 | 132 | 5 | 0 | 7 | 5 | 1 | 2 | 0 | 427 |
| NNPS | 1 | 0 | 110 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 142 |
| RB | 72 | 21 | 7 | 0 | 0 | 16 | 138 | 1 | 0 | 0 | 0 | 295 |
| RP | 0 | 0 | 0 | 0 | 39 | 0 | 65 | 0 | 0 | 0 | 0 | 104 |
| IN | 11 | 0 | 1 | 0 | 169 | 103 | 0 | 1 | 0 | 0 | 0 | 323 |
| VB | 17 | 64 | 9 | 0 | 2 | 0 | 1 | 0 | 4 | 7 | 85 | 189 |
| VBD | 10 | 5 | 3 | 0 | 0 | 0 | 0 | 3 | 0 | 143 | 2 | 166 |
| VCN | 101 | 3 | 3 | 0 | 0 | 0 | 0 | 3 | 108 | 0 | 1 | 221 |
| VBP | 5 | 34 | 3 | 1 | 1 | 0 | 2 | 49 | 6 | 3 | 0 | 104 |
| Total | 626 | 536 | 348 | 144 | 317 | 122 | 279 | 102 | 140 | 269 | 108 | 3651 |

NN/JJ NN

official knowledge

VBD RP/IN DT NN

made up the story

RB VBD/VBN NNS

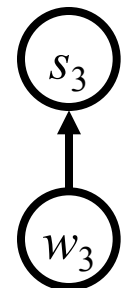
recently sold shares

Richer Features



Better Features

- Can do surprisingly well just looking at a word by itself:
 - Word the: the → DT
 - Lowercased word Importantly: importantly → RB
 - Prefixes unfathomable: un- → JJ
 - Suffixes Surprisingly: -ly → RB
 - Capitalization Meridian: CAP → NNP
 - Word shapes 35-year: d-x → JJ
- Then build a maxent (or whatever) model to predict tag
- Maxent $P(t|w)$: 93.7% / 82.6%





Why Linear Context is Useful

- Lots of rich local information!

RB
PRP VBD IN RB IN PRP VBD .
They left as soon as he arrived .

- We could fix this with a feature that looked at the next word

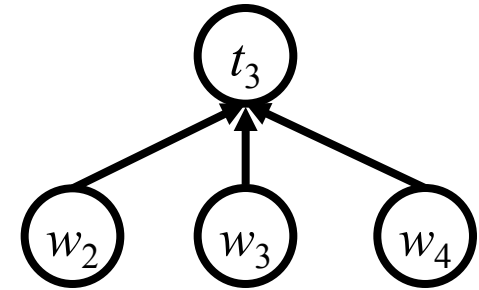
JJ
NNP NNS VBD VBN .
Intrinsic flaws remained undetected .

- We could fix this by linking capitalized words to their lowercase versions
- Solution: discriminative sequence models (MEMMs, CRFs)
- Reality check:
 - Taggers are already pretty good on newswire text...
 - What the world needs is taggers that work on other text!



Sequence-Free Tagging?

- What about looking at a word and its environment, but no sequence information?
 - Add in previous / next word the ____
 - Previous / next word shapes X ____ X
 - Occurrence pattern features [X: x X occurs]
 - Crude entity detection ____ (Inc. | Co.)
 - Phrasal verb in sentence? put ____
 - Conjunctions of these things
- All features except sequence: 96.6% / 86.8%
- Uses lots of features: > 200K
- Why isn't this the standard approach?





Named Entity Recognition

- Other sequence tasks use similar models
- Example: name entity recognition (NER)

PER PER O O O O O O ORG O O O O O O LOC LOC O

Tim Boon has signed a contract extension with Leicestershire which will keep him at Grace Road .

Local Context

| | Prev | Cur | Next |
|-------|-------|-------|------|
| State | Other | ??? | ??? |
| Word | at | Grace | Road |
| Tag | IN | NNP | NNP |
| Sig | x | Xx | Xx |



MEMM Taggers

- Idea: left-to-right local decisions, condition on previous tags and also entire input

$$P(\mathbf{t}|\mathbf{w}) = \prod_i P_{ME}(t_i|\mathbf{w}, t_{i-1}, t_{i-2})$$

- Train up $P(t_i|\mathbf{w}, t_{i-1}, t_{i-2})$ as a normal maxent model, then use to score sequences
 - This is referred to as an MEMM tagger [Ratnaparkhi 96]
 - Beam search effective! (Why?)
 - What about beam size 1?
-
- Subtle issues with local normalization (cf. Lafferty et al 01)



NER Features

Because of regularization term, the more common prefixes have larger weights even though entire-word features are more specific.

Local Context

| | Prev | Cur | Next |
|-------|-------|-------|------|
| State | Other | ??? | ??? |
| Word | at | Grace | Road |
| Tag | IN | NNP | NNP |
| Sig | x | Xx | Xx |

Feature Weights

| Feature Type | Feature | PERS | LOC |
|----------------------|---------|-------|-------|
| Previous word | at | -0.73 | 0.94 |
| Current word | Grace | 0.03 | 0.00 |
| Beginning bigram | <G | 0.45 | -0.04 |
| Current POS tag | NNP | 0.47 | 0.45 |
| Prev and cur tags | IN NNP | -0.10 | 0.14 |
| Previous state | Other | -0.70 | -0.92 |
| Current signature | Xx | 0.80 | 0.46 |
| Prev state, cur sig | O-Xx | 0.68 | 0.37 |
| Prev-cur-next sig | x-Xx-Xx | -0.69 | 0.37 |
| P. state - p-cur sig | O-x-Xx | -0.20 | 0.82 |
| ... | | | |
| Total: | | -0.58 | 2.68 |



Decoding

- Decoding MEMM taggers:

- Just like decoding HMMs, different local scores
- Viterbi, beam search, posterior decoding

- Viterbi algorithm (HMMs):

$$\delta_i(s) = \arg \max_{s'} P(s|s') P(w_{i-1}|s') \delta_{i-1}(s')$$

- Viterbi algorithm (MEMMs):

$$\delta_i(s) = \arg \max_{s'} P(s|s', \mathbf{w}) \delta_{i-1}(s')$$

- General:

$$\delta_i(s) = \arg \max_{s'} \phi_i(s', s) \delta_{i-1}(s')$$

Conditional Random Fields (and Friends)



Maximum Entropy II

- Remember: maximum entropy objective

$$L(\mathbf{w}) = \sum_i \left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i) - \log \sum_{\mathbf{y}} \exp(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y})) \right)$$

- Problem: lots of features allow perfect fit to training set
- Regularization (compare to smoothing)

$$\max_{\mathbf{w}} \sum_i \left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i) - \log \sum_{\mathbf{y}} \exp(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y})) \right) - k \|\mathbf{w}\|^2$$



Derivative for Maximum Entropy

$$L(\mathbf{w}) = -k\|\mathbf{w}\|^2 + \sum_i \left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i) - \log \sum_{\mathbf{y}} \exp(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y})) \right)$$

$$\frac{\partial L(\mathbf{w})}{\partial \mathbf{w}_n} = -2k\mathbf{w}_n + \sum_i \left(\mathbf{f}_i(\mathbf{y}^i)_n - \sum_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}_i) \mathbf{f}_i(\mathbf{y})_n \right)$$

Big weights are bad

Total count of feature n
in correct candidates

Expected count of
feature n in predicted
candidates



Global Discriminative Taggers

- Newer, higher-powered discriminative sequence models
 - CRFs (also perceptrons, M3Ns)
 - Do not decompose training into independent local regions
 - Can be deathly slow to train – require repeated inference on training set
- Differences tend not to be too important for POS tagging
- Differences more substantial on other sequence tasks
- However: one issue worth knowing about in local models
 - “Label bias” and other explaining away effects
 - MEMM taggers’ local scores can be near one without having both good “transitions” and “emissions”
 - This means that often evidence doesn’t flow properly
 - Why isn’t this a big deal for POS tagging?
 - Also: in decoding, condition on predicted, not gold, histories



Perceptron Taggers

- Linear models:

$$\text{score}(\mathbf{t}|\mathbf{w}) = \lambda^\top f(\mathbf{t}, \mathbf{w})$$

- ... that decompose along the sequence

$$= \lambda^\top \sum_i f(t_i, t_{i-1}, \mathbf{w}, i)$$

- ... allow us to predict with the Viterbi algorithm

$$\mathbf{t}^* = \arg \max_{\mathbf{t}} \text{score}(\mathbf{t}|\mathbf{w})$$

- ... which means we can train with the perceptron algorithm (or related updates, like MIRA)



Conditional Random Fields

- Make a maxent model over entire taggings

- MEMM

$$P(\mathbf{t}|\mathbf{w}) = \prod_i \frac{1}{Z(i)} \exp \left(\lambda^\top f(t_i, t_{i-1}, \mathbf{w}, i) \right)$$

- CRF

$$\begin{aligned} P(\mathbf{t}|\mathbf{w}) &= \frac{1}{Z(\mathbf{w})} \exp \left(\lambda^\top f(\mathbf{t}, \mathbf{w}) \right) \\ &= \frac{1}{Z(\mathbf{w})} \exp \left(\lambda^\top \sum_i f(t_i, t_{i-1}, \mathbf{w}, i) \right) \\ &= \frac{1}{Z(\mathbf{w})} \prod_i \phi_i(t_i, t_{i-1}) \end{aligned}$$



CRFs

- Like any maxent model, derivative is:

$$\frac{\partial L(\lambda)}{\partial \lambda} = \sum_k \left(\mathbf{f}_k(\mathbf{t}^k) - \sum_{\mathbf{t}} P(\mathbf{t}|\mathbf{w}_k) \mathbf{f}_k(\mathbf{t}) \right)$$

- So all we need is to be able to compute the expectation of each feature (for example the number of times the label pair *DT-NN* occurs, or the number of times *NN-interest* occurs) **under the model distribution**
- Critical quantity: counts of posterior marginals:

$$\text{count}(w, s) = \sum_{i:w_i=w} P(t_i = s|\mathbf{w})$$

$$\text{count}(s \rightarrow s') = \sum_i P(t_{i-1} = s, t_i = s'|\mathbf{w})$$

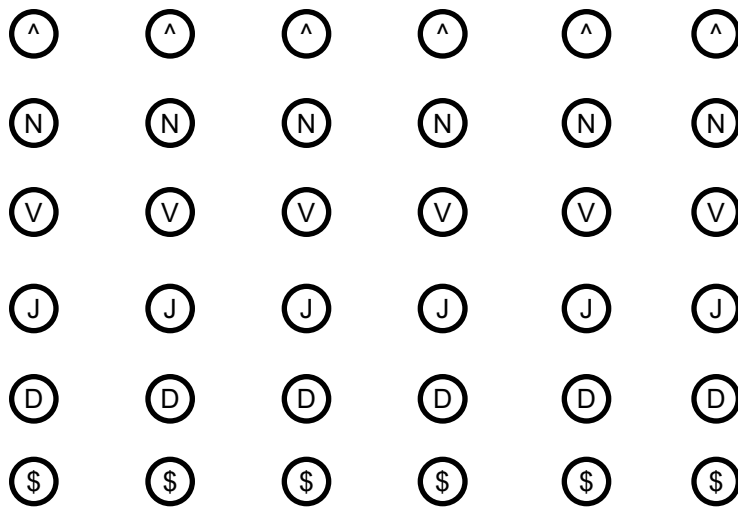


Computing Posterior Marginals

- How many (expected) times is word w tagged with s ?

$$\text{count}(w, s) = \sum_{i: w_i = w} P(t_i = s | \mathbf{w})$$

- How to compute that marginal?



START Fed raises interest rates END

$$\alpha_i(s) = \sum_{s'} \phi_i(s', s) \alpha_{i-1}(s')$$

$$\beta_i(s) = \sum_{s'} \phi_{i+1}(s, s') \beta_{i+1}(s')$$

$$P(t_i = s | \mathbf{w}) = \frac{\alpha_i(s) \beta_i(s)}{\alpha_N(\text{END})}$$



Transformation-Based Learning

- [Brill 95] presents a *transformation-based* tagger

- Label the training set with most frequent tags

DT MD VBD VBD .

The can was rusted .

- Add transformation rules which reduce training mistakes

- MD → NN : DT __

- VBD → VBN : VBD __ .

- Stop when no transformations do sufficient good
 - Does this remind anyone of anything?

- Probably the most widely used tagger (esp. outside NLP)
- ... but definitely not the most accurate: 96.6% / 82.0 %



Learned Transformations

■ What gets learned? [from Brill 95]

| Change Tag | | | |
|------------|------|-----|--|
| # | From | To | Condition |
| 1 | NN | VB | Previous tag is <i>TO</i> |
| 2 | VBP | VB | One of the previous three tags is <i>MD</i> |
| 3 | NN | VB | One of the previous two tags is <i>MD</i> |
| 4 | VB | NN | One of the previous two tags is <i>DT</i> |
| 5 | VBD | VBN | One of the previous three tags is <i>VBZ</i> |
| 6 | VBN | VBD | Previous tag is <i>PRP</i> |
| 7 | VBN | VBD | Previous tag is <i>NNP</i> |
| 8 | VBD | VBN | Previous tag is <i>VBD</i> |
| 9 | VBP | VB | Previous tag is <i>TO</i> |
| 10 | POS | VBZ | Previous tag is <i>PRP</i> |
| 11 | VB | VBP | Previous tag is <i>NNS</i> |
| 12 | VBD | VBN | One of previous three tags is <i>VBP</i> |
| 13 | IN | WDT | One of next two tags is <i>VB</i> |
| 14 | VBD | VBN | One of previous two tags is <i>VB</i> |
| 15 | VB | VBP | Previous tag is <i>PRP</i> |
| 16 | IN | WDT | Next tag is <i>VBZ</i> |
| 17 | IN | DT | Next tag is <i>NN</i> |
| 18 | JJ | NNP | Next tag is <i>NNP</i> |
| 19 | IN | WDT | Next tag is <i>VBD</i> |
| 20 | JJR | RBR | Next tag is <i>JJ</i> |

| Change Tag | | | |
|------------|------|-----|--|
| # | From | To | Condition |
| 1 | NN | NNS | Has suffix -s |
| 2 | NN | CD | Has character . |
| 3 | NN | JJ | Has character - |
| 4 | NN | VBN | Has suffix -ed |
| 5 | NN | VBG | Has suffix -ing |
| 6 | ?? | RB | Has suffix -ly |
| 7 | ?? | JJ | Adding suffix -ly results in a word. |
| 8 | NN | CD | The word \$ can appear to the left. |
| 9 | NN | JJ | Has suffix -al |
| 10 | NN | VB | The word would can appear to the left. |
| 11 | NN | CD | Has character 0 |
| 12 | NN | JJ | The word be can appear to the left. |
| 13 | NNS | JJ | Has suffix -us |
| 14 | NNS | VBZ | The word it can appear to the left. |
| 15 | NN | JJ | Has suffix -ble |
| 16 | NN | JJ | Has suffix -ic |
| 17 | NN | CD | Has character 1 |
| 18 | NNS | NN | Has suffix -ss |
| 19 | ?? | JJ | Deleting the prefix un- results in a word |
| 20 | NN | JJ | Has suffix -ive |



EngCG Tagger

- English constraint grammar tagger
 - [Tapanainen and Voutilainen 94]
 - Something else you should know about
 - Hand-written and knowledge driven
 - “Don’t guess if you know” (general point about modeling more structure!)
 - Tag set doesn’t make all of the hard distinctions as the standard tag set (e.g. JJ/NN)
 - They get stellar accuracies: 99% on *their* tag set
 - Linguistic representation matters...
 - ... but it’s easier to win when you make up the rules

```
walk
walk <SV> <SVO> V SUBJUNCTIVE VFIN
walk <SV> <SVO> V IMP VFIN
walk <SV> <SVO> V INF
walk <SV> <SVO> V PRES -SG3 VFIN
walk N NOM SG
```

```
walk V-SUBJUNCTIVE V-IMP V-INF
V-PRES-BASE N-NOM-SG
```



Domain Effects

- Accuracies degrade outside of domain
 - Up to triple error rate
 - Usually make the most errors on the things you care about in the domain (e.g. protein names)
- Open questions
 - How to effectively exploit unlabeled data from a new domain (what could we gain?)
 - How to best incorporate domain lexica in a principled way (e.g. UMLS specialist lexicon, ontologies)

Unsupervised Tagging



Unsupervised Tagging?

- AKA part-of-speech induction
- Task:
 - Raw sentences in
 - Tagged sentences out
- Obvious thing to do:
 - Start with a (mostly) uniform HMM
 - Run EM
 - Inspect results



EM for HMMs: Process

- Alternate between recomputing distributions over hidden variables (the tags) and reestimating parameters
- Crucial step: we want to tally up how many (fractional) counts of each kind of transition and emission we have under current params:

$$\text{count}(w, s) = \sum_{i:w_i=w} P(t_i = s | \mathbf{w})$$

$$\text{count}(s \rightarrow s') = \sum_i P(t_{i-1} = s, t_i = s' | \mathbf{w})$$

- Same quantities we needed to train a CRF!



EM for HMMs: Quantities

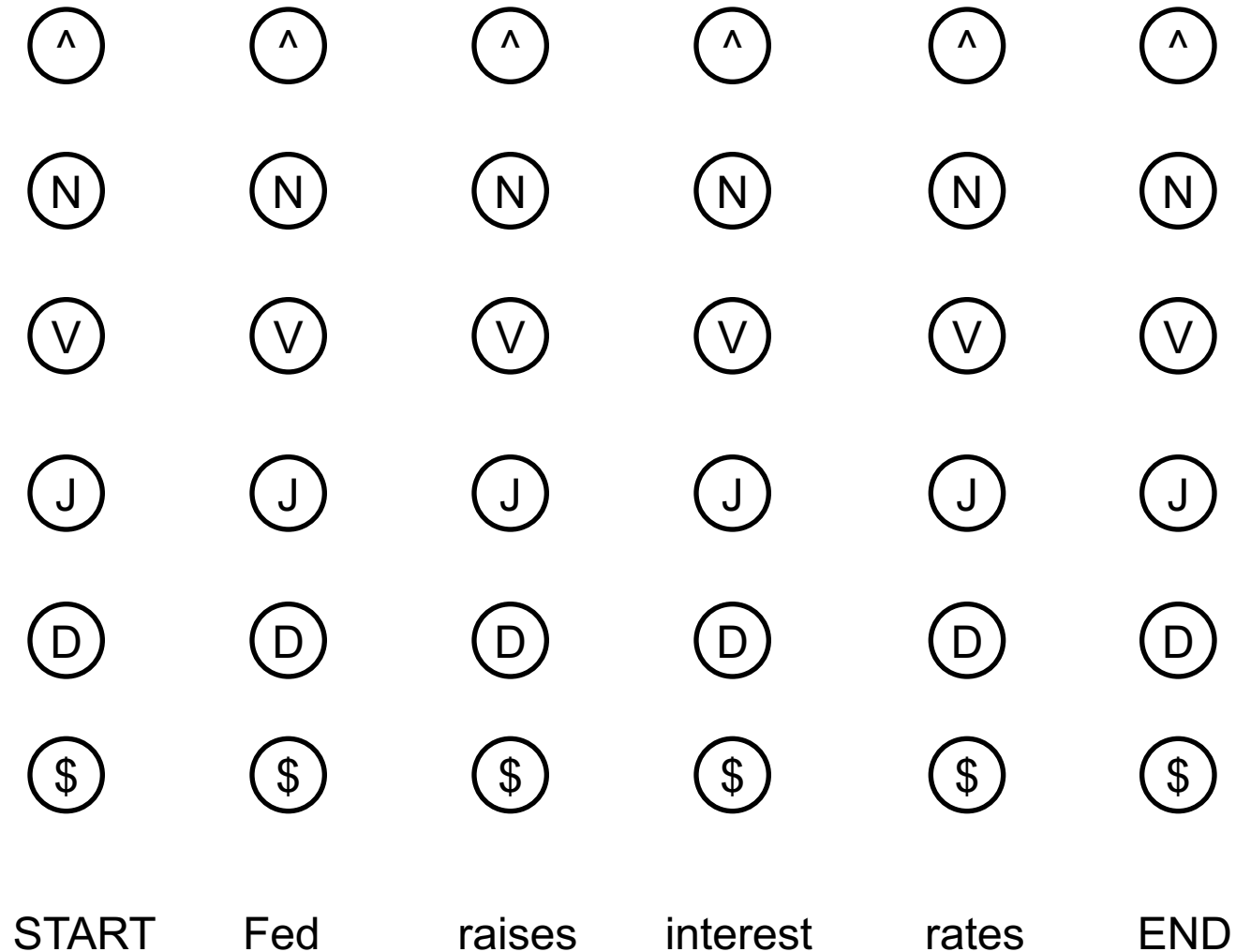
- Total path values (correspond to probabilities here):

$$\begin{aligned}\alpha_i(s) &= P(w_0 \dots w_i, s_i) \\ &= \sum_{s_{i-1}} P(s_i | s_{i-1}) P(w_i | s_i) \alpha_{i-1}(s_{i-1})\end{aligned}$$

$$\begin{aligned}\beta_i(s) &= P(w_{i+1} \dots w_n | s_i) \\ &= \sum_{s_{i+1}} P(s_{i+1} | s_i) P(w_{i+1} | s_{i+1}) \beta_{i+1}(s_{i+1})\end{aligned}$$



The State Lattice / Trellis





EM for HMMs: Process

- From these quantities, can compute expected transitions:

$$\text{count}(s \rightarrow s') = \frac{\sum_i \alpha_i(s) P(s'|s) P(w_i|s) \beta_{i+1}(s')}{P(\mathbf{w})}$$

- And emissions:

$$\text{count}(w, s) = \frac{\sum_{i:w_i=w} \alpha_i(s) \beta_{i+1}(s)}{P(\mathbf{w})}$$



Merialdo: Setup

- Some (discouraging) experiments [Merialdo 94]
- Setup:
 - You know the set of allowable tags for each word
 - Fix k training examples to their true labels
 - Learn $P(w|t)$ on these examples
 - Learn $P(t|t_{-1}, t_{-2})$ on these examples
 - On n examples, re-estimate with EM
- Note: we know allowed tags but not frequencies



Merialdo: Results

| Number of tagged sentences used for the initial model | | | | | | | |
|---|---|------|------|------|-------|-------|------|
| | 0 | 100 | 2000 | 5000 | 10000 | 20000 | all |
| Iter | Correct tags (% words) after ML on 1M words | | | | | | |
| 0 | 77.0 | 90.0 | 95.4 | 96.2 | 96.6 | 96.9 | 97.0 |
| 1 | 80.5 | 92.6 | 95.8 | 96.3 | 96.6 | 96.7 | 96.8 |
| 2 | 81.8 | 93.0 | 95.7 | 96.1 | 96.3 | 96.4 | 96.4 |
| 3 | 83.0 | 93.1 | 95.4 | 95.8 | 96.1 | 96.2 | 96.2 |
| 4 | 84.0 | 93.0 | 95.2 | 95.5 | 95.8 | 96.0 | 96.0 |
| 5 | 84.8 | 92.9 | 95.1 | 95.4 | 95.6 | 95.8 | 95.8 |
| 6 | 85.3 | 92.8 | 94.9 | 95.2 | 95.5 | 95.6 | 95.7 |
| 7 | 85.8 | 92.8 | 94.7 | 95.1 | 95.3 | 95.5 | 95.5 |
| 8 | 86.1 | 92.7 | 94.6 | 95.0 | 95.2 | 95.4 | 95.4 |
| 9 | 86.3 | 92.6 | 94.5 | 94.9 | 95.1 | 95.3 | 95.3 |
| 10 | 86.6 | 92.6 | 94.4 | 94.8 | 95.0 | 95.2 | 95.2 |



Distributional Clustering

◆ *the president said that the downturn was over* ◆

| | |
|------------------|---------------------------|
| <i>president</i> | <i>the ___ of</i> |
| <i>president</i> | <i>the ___ said</i> ← |
| <i>governor</i> | <i>the ___ of</i> |
| <i>governor</i> | <i>the ___ appointed</i> |
| <i>said</i> | <i>sources ___</i> ◆ |
| <i>said</i> | <i>president ___ that</i> |
| <i>reported</i> | <i>sources ___</i> ◆ |

*president
governor*

*said
reported*

*the
a*

[Finch and Chater 92, Shuetze 93, many others]



Distributional Clustering

- Three main variants on the same idea:
 - Pairwise similarities and heuristic clustering
 - E.g. [Finch and Chater 92]
 - Produces dendrograms
 - Vector space methods
 - E.g. [Shuetze 93]
 - Models of ambiguity
 - Probabilistic methods
 - Various formulations, e.g. [Lee and Pereira 99]

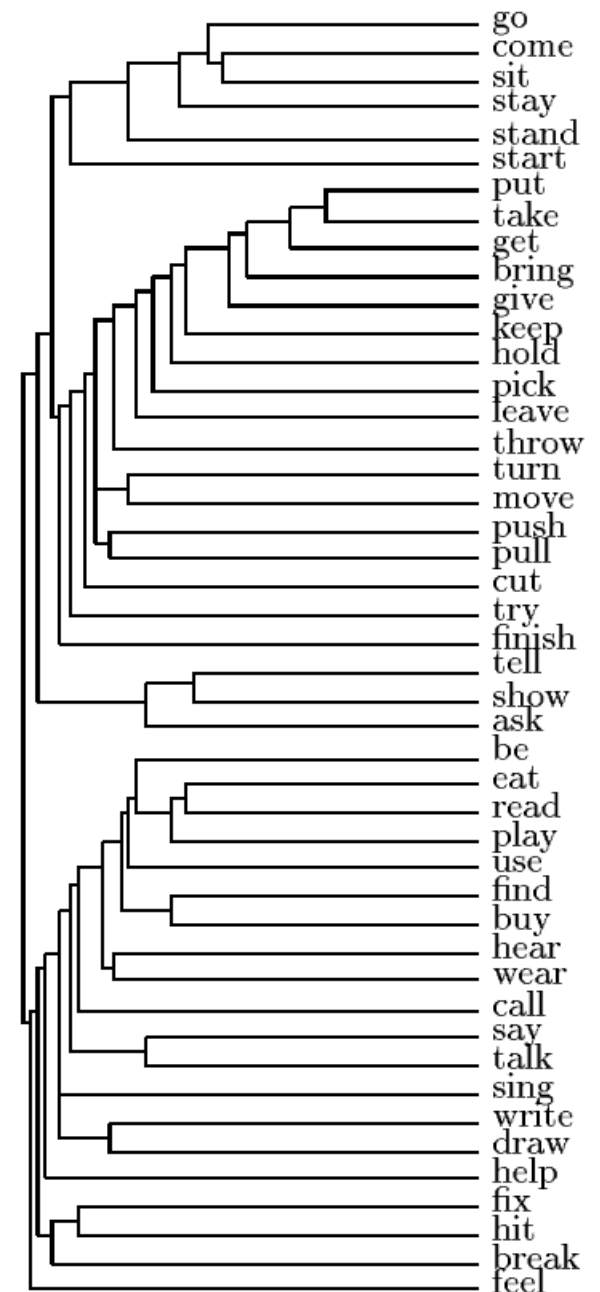
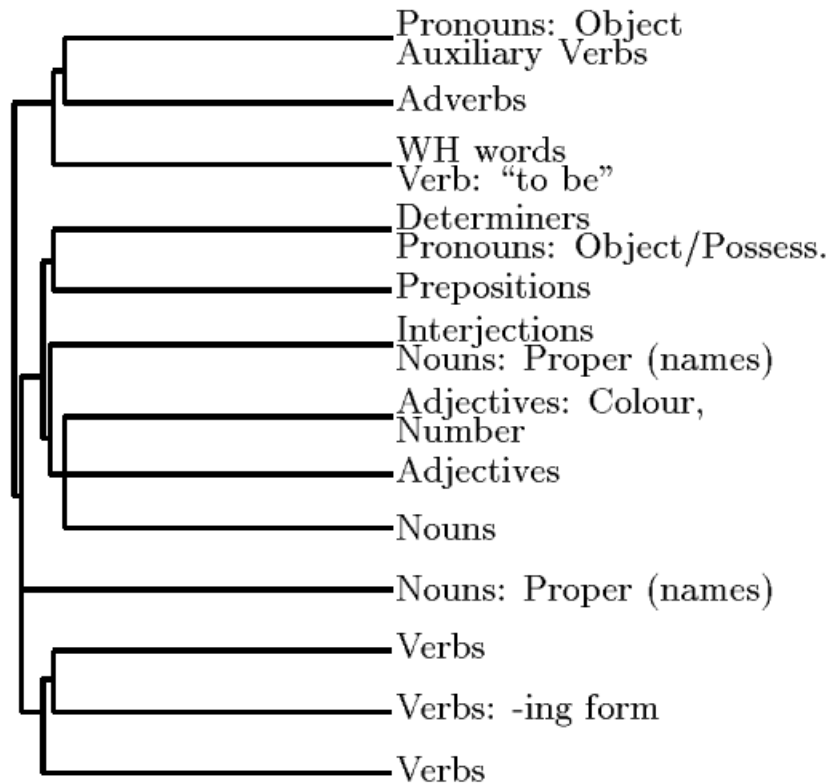


Nearest Neighbors

| word | nearest neighbors |
|-------------|---|
| accompanied | submitted banned financed developed authorized headed canceled awarded barred |
| almost | virtually merely formally fully quite officially just nearly only less |
| causing | reflecting forcing providing creating producing becoming carrying particularly |
| classes | elections courses payments losses computers performances violations levels pictures |
| directors | professionals investigations materials competitors agreements papers transactions |
| goal | mood roof eye image tool song pool scene gap voice |
| japanese | chinese iraqi american western arab foreign european federal soviet indian |
| represent | reveal attend deliver reflect choose contain impose manage establish retain |
| think | believe wish know realize wonder assume feel say mean bet |
| york | angeles francisco sox rouge kong diego zone vegas inning layer |
| on | through in at over into with from for by across |
| must | might would could cannot will should can may does helps |
| they | we you i he she nobody who it everybody there |

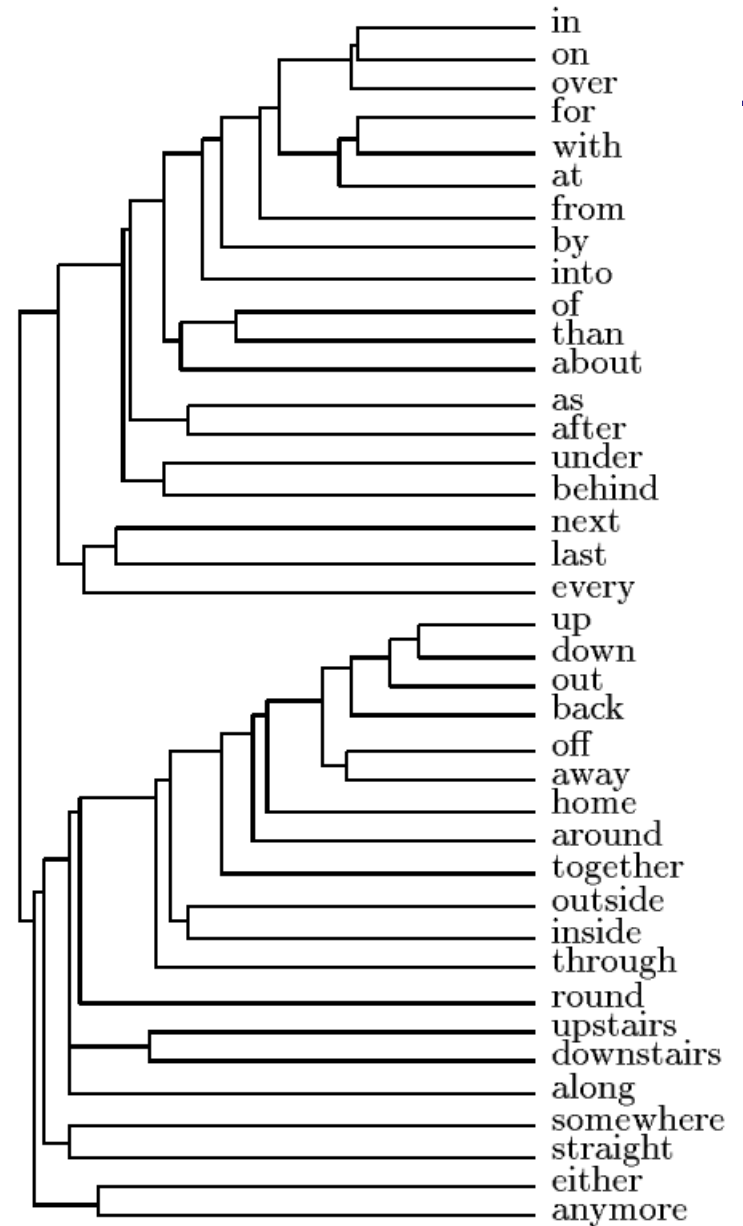
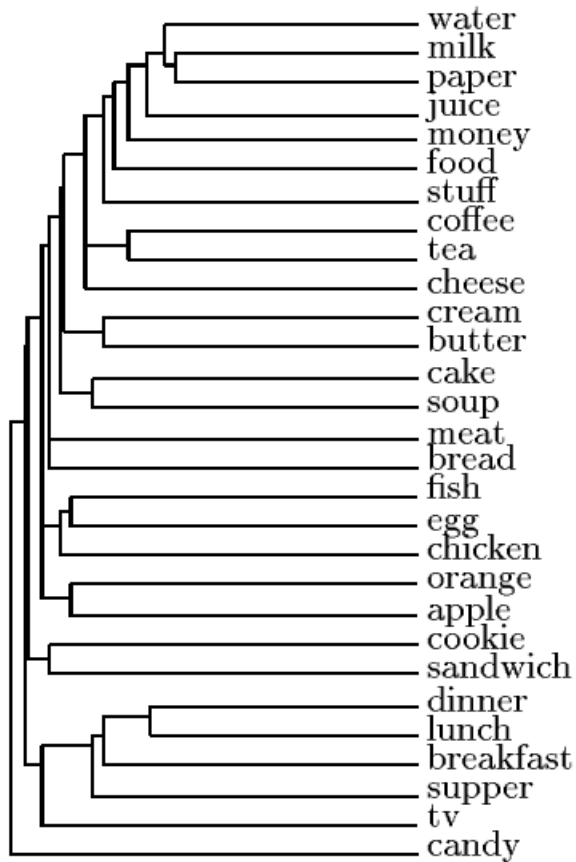


Dendrograms





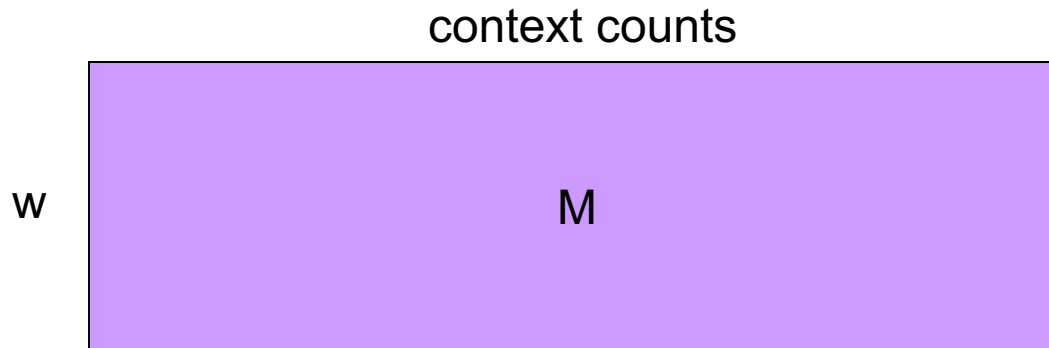
Dendrograms



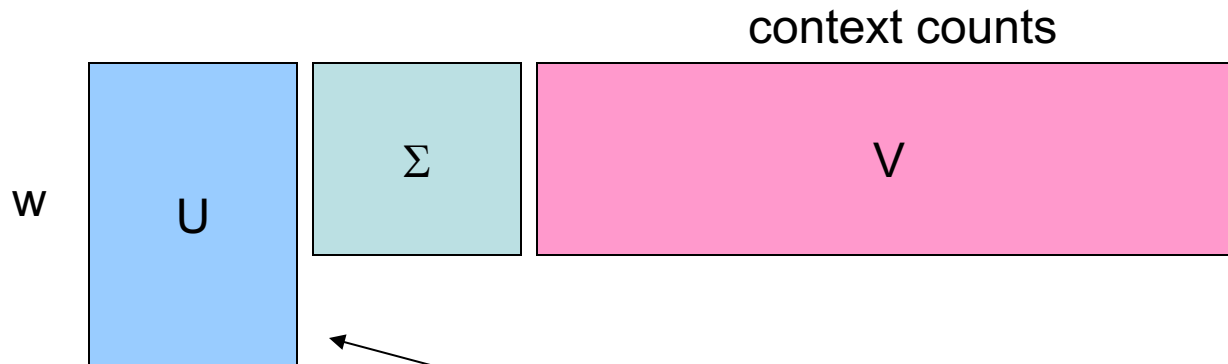


Vector Space Version

- [Shuetze 93] clusters words as points in \mathbb{R}^n



- Vectors too sparse, use SVD to reduce

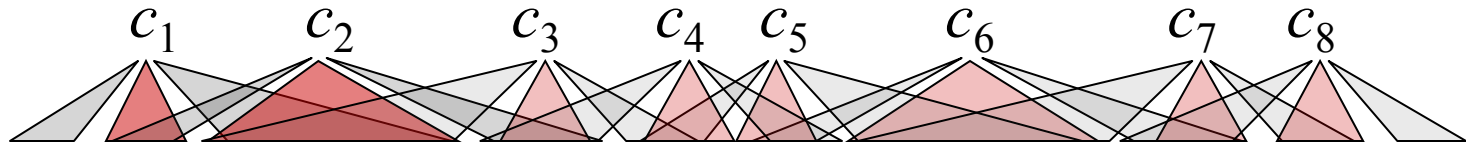


Cluster these 50-200 dim vectors instead.

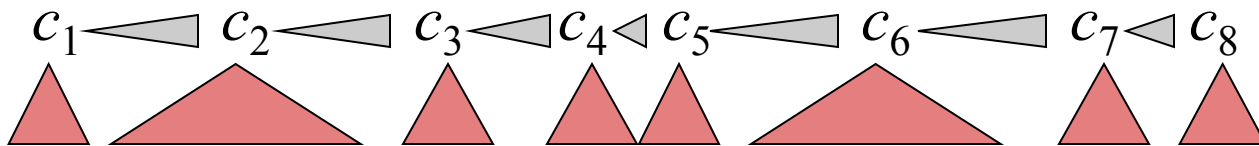


A Probabilistic Version?

$$P(S, C) = \prod_i P(c_i) P(w_i | c_i) P(w_{i-1}, w_{i+1} | c_i)$$



◆ *the president said that the downturn was over* ◆



◆ *the president said that the downturn was over* ◆



What Else?

- Various newer ideas:

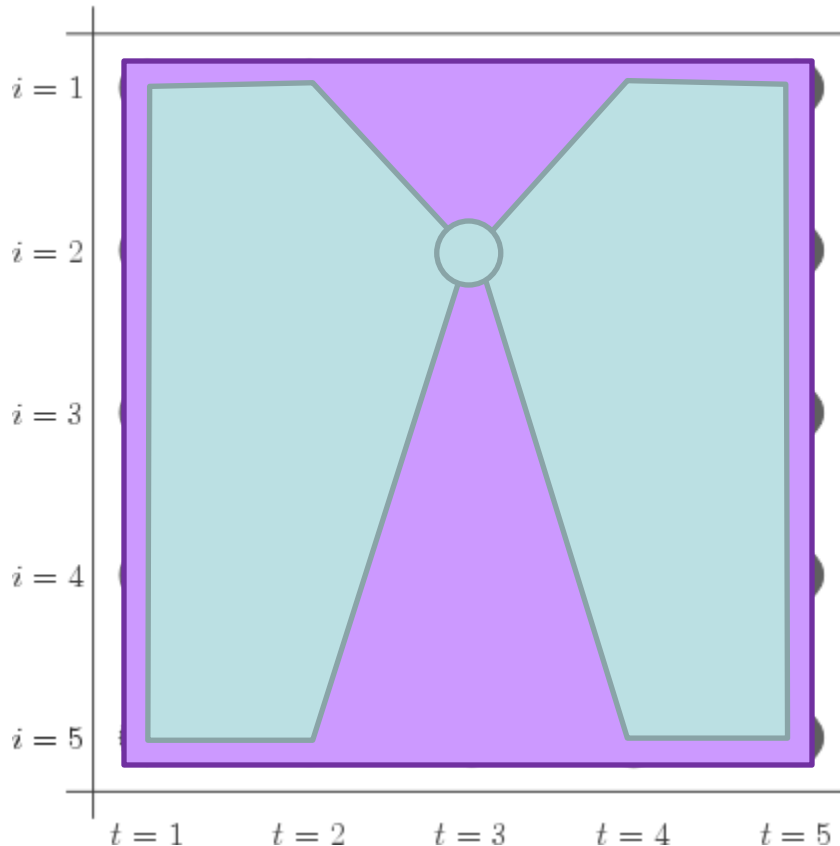
- Context distributional clustering [Clark 00]
- Morphology-driven models [Clark 03]
- Contrastive estimation [Smith and Eisner 05]
- Feature-rich induction [Haghighi and Klein 06]

- Also:

- What about ambiguous words?
- Using wider context signatures has been used for learning synonyms (what's wrong with this approach?)
- Can extend these ideas for grammar induction (later)



Computing Marginals

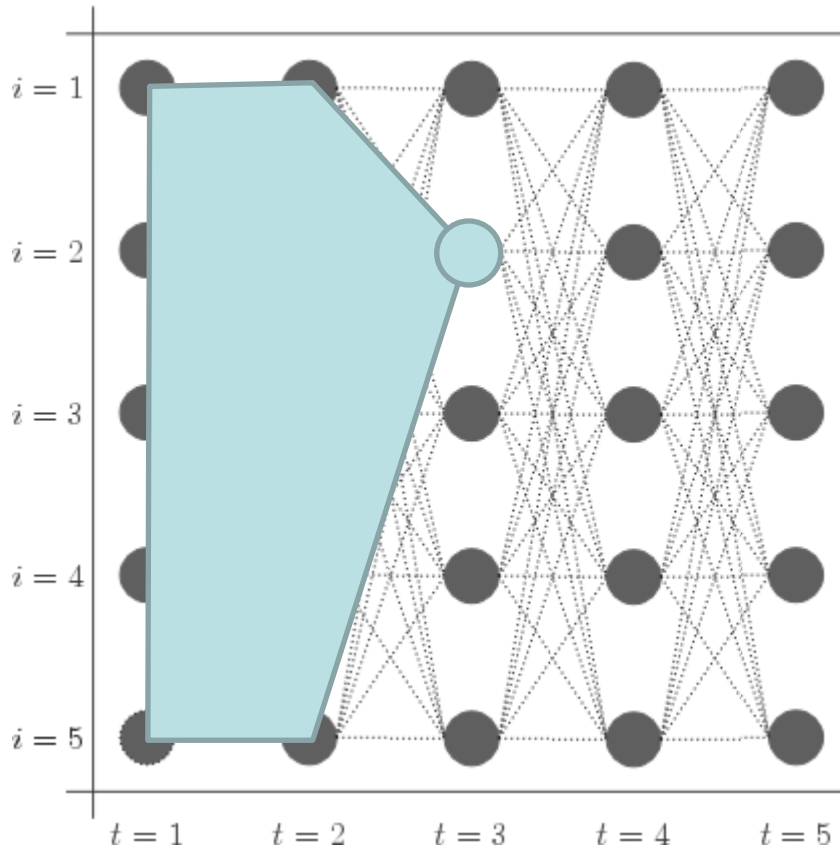


$$P(s_t|x) = \frac{P(s_t, x)}{P(x)}$$

= sum of all paths through s at t
sum of all paths



Forward Scores

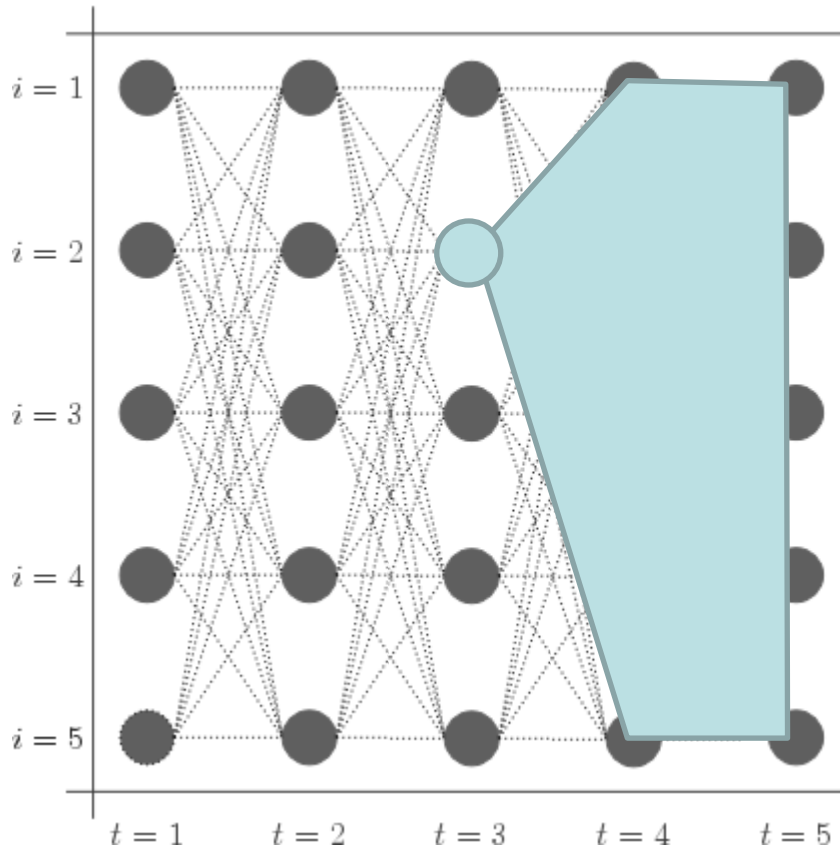


$$v_t(s_t) = \max_{s_{t-1}} v_{t-1}(s_{t-1}) \phi_t(s_{t-1}, s_t)$$

$$\alpha_t(s_t) = \sum_{s_{t-1}} \alpha_{t-1}(s_{t-1}) \phi_t(s_{t-1}, s_t)$$



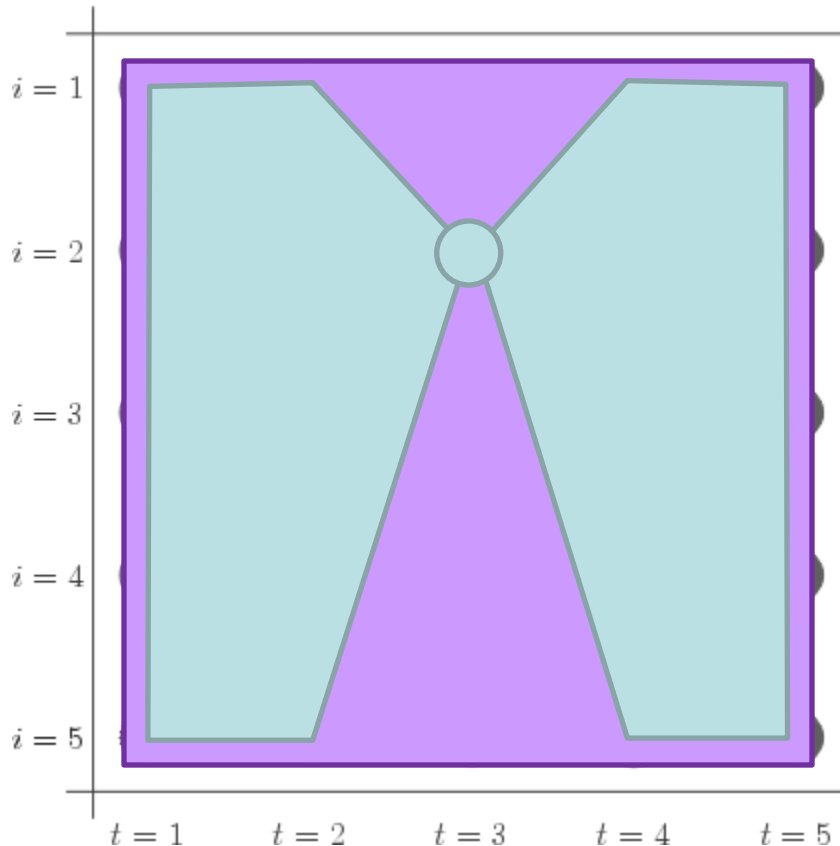
Backward Scores



$$\beta_t(s_t) = \sum_{s_{t+1}} \beta_{t+1}(s_{t+1}) \phi_t(s_t, s_{t+1})$$



Total Scores



$$P(s_t, x) = \alpha_t(s_t)\beta_t(s_t)$$

$$P(x) = \sum_{s_t} \alpha_t(s_t)\beta_t(s_t)$$

$$= \alpha_T(\text{stop})$$

$$= \beta_0(\text{start})$$