

# Statistical NLP

## Spring 2008



### Lecture 8: Word Classes

Dan Klein – UC Berkeley

## What's Next for POS Tagging

- Better features!

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: maximum entropy sequence models

- Reality check:

- Taggers are already pretty good on WSJ journal text...
- What the world needs is taggers that work on other text!
- Also: same techniques used for other sequence models (NER, etc)

## Maxent Taggers

- MEMMs: use local discriminative models

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

$$\frac{1}{Z} \exp(w^T f(t_i, t_{i-1}, t_{i-2}, w, i))$$

- Train up  $P(t_i|w, t_{i-1}, t_{i-2}, i)$  as a normal maxent problem, then use to score sequences
- Referred to as a *maxent tagger* [Ratnaparkhi 96]
- Beam search effective! (Why?)
- What's the advantage of beam size 1?

## Feature Templates

- Important distinction:

- Features:  $\langle w_0 = \text{future}, t_0 = \text{JJ} \rangle$
- Feature templates:  $\langle w_0, t_0 \rangle$

- In maxent taggers:

- Can now add *edge* feature templates:
  - $\langle t_1, t_0 \rangle$
  - $\langle t_2, t_1, t_0 \rangle$
- Also, mixed feature templates:
  - $\langle t_1, w_0, t_0 \rangle$

## Decoding

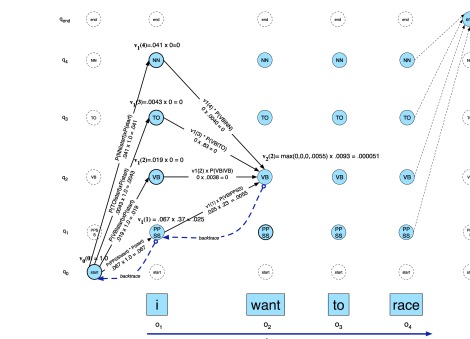
- Decoding maxent taggers:
  - Just like decoding HMMs
  - Viterbi, beam search, posterior decoding
- Viterbi algorithm (HMMs):

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

- Viterbi algorithm (Maxent):

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

## HMM Trellis



## TBL Tagger

- [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 not the most accurate: 96.6% / 82.0 %

## TBL Tagger II

- What gets learned? [from Brill 95]

#	Change	Tag	Condition
1	NN	VB	Previous tag is <i>TG</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>VBE</i>
6	VBN	VBD	Previous tag is <i>PREP</i>
7	VBN	VBD	Previous tag is <i>NVP</i>
8	VBD	VBN	Previous tag is <i>VBD</i>
9	VBP	VB	Previous tag is <i>TG</i>
10	POS	VBZ	Previous tag is <i>PREP</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>PREP</i>
16	IN	WDT	Next tag is <i>VBE</i>
17	IN	DT	Next tag is <i>IN</i>
18	JJ	NFP	Next tag is <i>NVP</i>
19	IN	WDT	Next tag is <i>VBD</i>
20	JR	RBR	Next tag is <i>JJ</i>

## 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: 98.5% on *their* tag set
  - Linguistic representation matters...
  - ... but it's easier to win when you make up the rules

## CRF Taggers

- Newer, higher-powered discriminative sequence models
  - CRFs (also voted 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
  - Maxent 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

## CRFs

- Make a maxent model over entire taggings

- MEMM

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

- CRF

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

## CRFs

- Like any maxent model, derivative is:

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

- So all we need is to be able to compute the expectation each feature, for example the number of times the label pair *DT-NN* occurs, or the number of times *NN-interest* occurs in a sentence

- How many times does, say, *DT-NN* occur at position 10? The ratio of the scores of trajectories with that configuration to the score of all

- This requires exactly the same forward-backward score ratios as for EM, but using the local potentials  $\phi$  instead of the local probabilities

## 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?

- 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

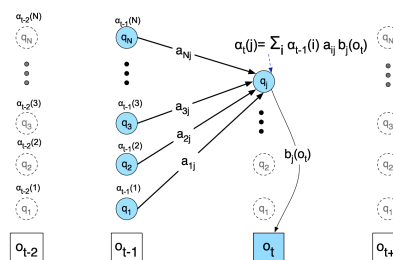
- Alternate between recomputing distributions over hidden variables (the tags) and reestimating parameters
- Crucial step: we want to tally up how many (fractional) counts of each kind of transition and emission we have under current params:

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

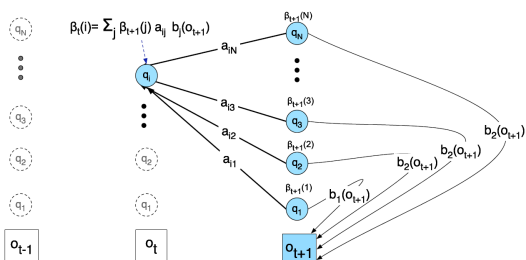
$$\text{count}(w, s) = \sum_{i: w_i = w} P(t_i = s | \mathbf{w})$$

- But we need a dynamic program to help, because there are too many sequences to sum over

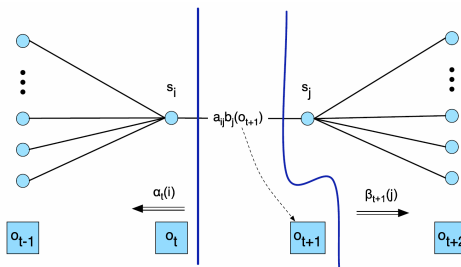
## Forward Recurrence



## Backward Recurrence



## Fractional Transitions



## EM for HMMs: Quantities

- Cache total path values:

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

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

- Can calculate in  $O(s^2n)$  time (why?)

## EM for HMMs: Process

- From these quantities, we can re-estimate transitions:

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

- And emissions:

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

- If you don't get these formulas immediately, just think about hard EM instead, where we re-estimate from the Viterbi sequences

## 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

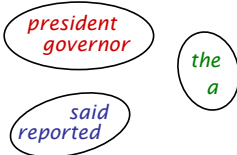
## Merialdo: Results

		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

president	the ___ of
president	the ___ said
governor	the ___ of
governor	the ___ appointed
said	sources ___
said	president ___ that
reported	sources ___



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

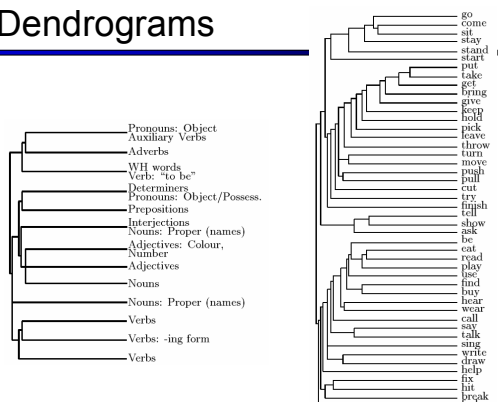
## Distributional Clustering

- 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]

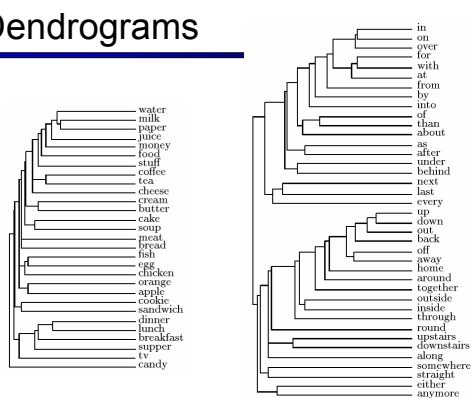
## Nearest Neighbors

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	singles 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

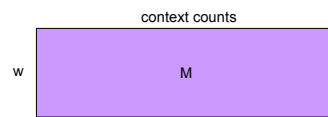


## Dendrograms

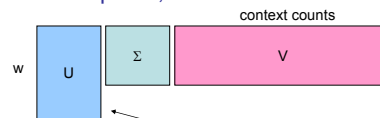


## Vector Space Version

- [Shuetze 93] clusters words as points in  $R^n$

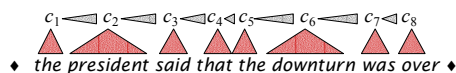
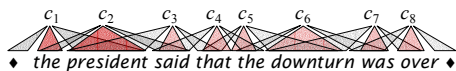


- Vectors too sparse, use SVD to reduce



## A Probabilistic Version?

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



## What Else?

- Various newer ideas:
  - Context distributional clustering [Clark 00]
  - Morphology-driven models [Clark 03]
  - Contrastive estimation [Smith and Eisner 05]
- 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)