Statistical NLP
Spring 2007

Lecture 7: Word Classes

Dan Klein – UC Berkeley

What's Next for POS Tagging

- Better features!

  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: 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)

Common Errors

- Common errors [from Toutanova & Manning 00]

	JJ	NN	NNP	NNPS	RB	RP	IN	VB	VBD	VBN	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
VBN	**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

Sequence-Free Tagging?

- What about looking at a word and it's 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?

Maxent Taggers

- One step up: also condition on previous tags

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

- Train up P(ti|w,ti-1,ti-2,i) as a normal maxent problem, then use to score sequences
- This is 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: $<w_0=\text{future}, t_0=\text{JJ}>$
  - Feature templates: $<w_0, t_0>$
- In maxent taggers:
  - Can now add *edge* feature templates:
    - $< t_{-1}, t_0>$
    - $< t_{-2}, t_{-1}, t_0>$
  - Also, mixed feature templates:
    - $< t_{-1}, w_0 , t_0 >$

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):

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

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

	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: 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?

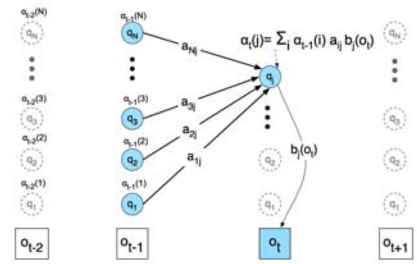
Domain Effects

- 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)

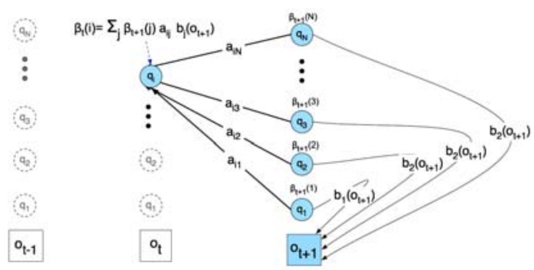
Unsupervised Tagging?

- 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

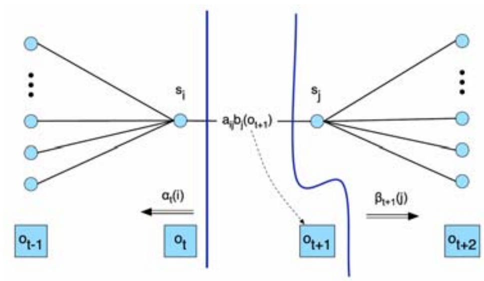
Forward Recurrence



Backward Recurrence



Fractional Transitions



EM for HMMs: Quantities

- Cache total path values:

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

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

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

EM for HMMs: Process

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

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

- And emissions:

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

- If you don't get these formulas immediately, just think about hard EM instead, where were 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 __ ♦*

president
governor

the
a

said
reported

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

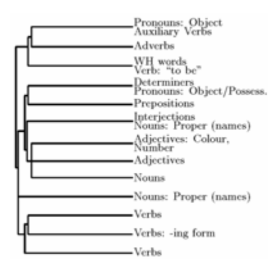
Distributional Clustering

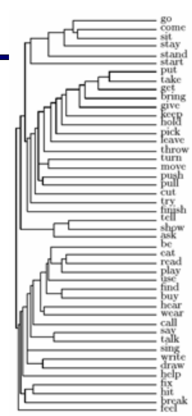
- Three main variants on the same idea:
  - Pairwise similarities and heuristic clustering
    - E.g. [Finch and Chater 92]
    - Produces dendrograms
  - Vector space methods
    - E.g. [Shuetze 93]
    - Models of ambiguity
  - Probabilistic methods
    - Various formulations, e.g. [Lee and Pereira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

Nearest Neighbors

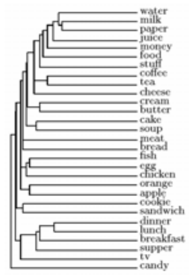
word	nearest neighbors
accompanied	submitted banned financed developed authorized headed canceled awarded barred
almost	virtually merely formally fully quite officially just nearly only less
causing	reflecting forcing providing creating producing becoming carrying particularly
classes	elections courses 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

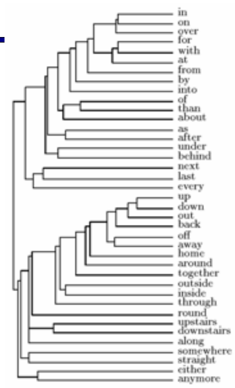
Dendrograms





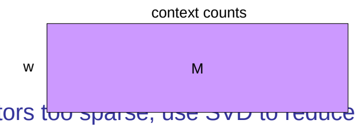
Dendrograms



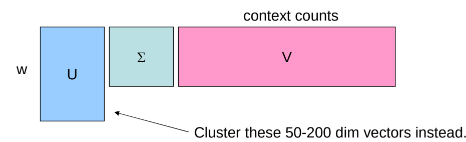


Vector Space Version

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

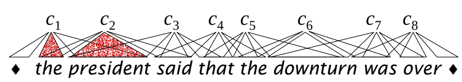


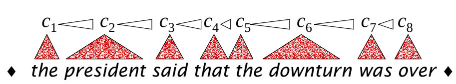
- Vectors too sparse, use SVD to reduce



A Probabilistic Version?

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





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)