What’s Next for POS Tagging

- Better features!

  They left as soon as he arrived.

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

  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

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

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

- Train up \( P(t|w, t_{i-1}, t_{i-2}, i) \) 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: \( <w_0=\text{future}, t_0=\text{JJ}> \)
  - 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):
  \[ \delta_i(s) = \arg \max_{s'} P(s|s', w, i) \delta_{i-1}(s') \]

---

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]

```latex
<table>
<thead>
<tr>
<th>Change Tag</th>
<th>From</th>
<th>To</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NN</td>
<td>VB</td>
<td>Previous tag is TO</td>
</tr>
<tr>
<td>2</td>
<td>VBP</td>
<td>VB</td>
<td>One of the previous two tags is MD</td>
</tr>
<tr>
<td>3</td>
<td>NN</td>
<td>VB</td>
<td>One of the previous two tags is MD</td>
</tr>
<tr>
<td>4</td>
<td>VB</td>
<td>NN</td>
<td>One of the previous three tags is DT</td>
</tr>
<tr>
<td>5</td>
<td>VBD</td>
<td>VBN</td>
<td>One of the previous three tags is VBD</td>
</tr>
<tr>
<td>6</td>
<td>VBN</td>
<td>VBD</td>
<td>Previous tag is PRP</td>
</tr>
<tr>
<td>7</td>
<td>VBN</td>
<td>VBD</td>
<td>Previous tag is XFP</td>
</tr>
<tr>
<td>8</td>
<td>VBD</td>
<td>VBN</td>
<td>Previous tag is VBD</td>
</tr>
<tr>
<td>9</td>
<td>VBP</td>
<td>VB</td>
<td>Previous tag is TO</td>
</tr>
<tr>
<td>10</td>
<td>POS</td>
<td>VBZ</td>
<td>Previous tag is PRP</td>
</tr>
<tr>
<td>11</td>
<td>VB</td>
<td>VBP</td>
<td>Previous tag is XFP</td>
</tr>
<tr>
<td>12</td>
<td>IN</td>
<td>WDT</td>
<td>One of next two tags is VB</td>
</tr>
<tr>
<td>13</td>
<td>VBD</td>
<td>VBN</td>
<td>One of previous three tags is VBP</td>
</tr>
<tr>
<td>14</td>
<td>VBD</td>
<td>VBN</td>
<td>One of previous two tags is VB</td>
</tr>
<tr>
<td>15</td>
<td>VBP</td>
<td>VBP</td>
<td>Previous tag is PRP</td>
</tr>
<tr>
<td>16</td>
<td>IN</td>
<td>WDT</td>
<td>Next tag is VBZ</td>
</tr>
<tr>
<td>17</td>
<td>IN</td>
<td>DT</td>
<td>Next tag is FAX</td>
</tr>
<tr>
<td>18</td>
<td>JJ</td>
<td>NNP</td>
<td>Next tag is NNP</td>
</tr>
<tr>
<td>19</td>
<td>IN</td>
<td>WDT</td>
<td>Next tag is VBD</td>
</tr>
<tr>
<td>20</td>
<td>JBR</td>
<td>RDR</td>
<td>Next tag in JJ</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Change Tag</th>
<th>From</th>
<th>To</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NN</td>
<td>NNS</td>
<td>Has suffix -s</td>
</tr>
<tr>
<td>2</td>
<td>NN</td>
<td>CD</td>
<td>Has character .</td>
</tr>
<tr>
<td>3</td>
<td>NN</td>
<td>JJ</td>
<td>Has character -</td>
</tr>
<tr>
<td>4</td>
<td>NN</td>
<td>VBN</td>
<td>Has suffix -ed</td>
</tr>
<tr>
<td>5</td>
<td>NNS</td>
<td>VBZ</td>
<td>Has suffix -ing</td>
</tr>
<tr>
<td>6</td>
<td>RB</td>
<td>JJ</td>
<td>Has suffix -ly</td>
</tr>
<tr>
<td>7</td>
<td>JJ</td>
<td>JJ</td>
<td>Adding suffix -ly results in a word.</td>
</tr>
<tr>
<td>8</td>
<td>NN</td>
<td>CD</td>
<td>The word # can appear to the left.</td>
</tr>
<tr>
<td>9</td>
<td>NN</td>
<td>JJ</td>
<td>Has suffix -al</td>
</tr>
<tr>
<td>10</td>
<td>NN</td>
<td>VB</td>
<td>The word would can appear to the right.</td>
</tr>
<tr>
<td>11</td>
<td>NN</td>
<td>CD</td>
<td>Has character #</td>
</tr>
<tr>
<td>12</td>
<td>NN</td>
<td>JJ</td>
<td>The word be can appear to the left.</td>
</tr>
<tr>
<td>13</td>
<td>NNS</td>
<td>JJ</td>
<td>Has suffix -ms</td>
</tr>
<tr>
<td>14</td>
<td>NNS</td>
<td>VBZ</td>
<td>The word it can appear to the left.</td>
</tr>
<tr>
<td>15</td>
<td>NN</td>
<td>JJ</td>
<td>Has suffix -ble</td>
</tr>
<tr>
<td>16</td>
<td>NN</td>
<td>JJ</td>
<td>Has suffix -le</td>
</tr>
<tr>
<td>17</td>
<td>NN</td>
<td>CD</td>
<td>Has character #</td>
</tr>
<tr>
<td>18</td>
<td>NNS</td>
<td>NN</td>
<td>Has suffix -ss</td>
</tr>
<tr>
<td>19</td>
<td>JJ</td>
<td>JJ</td>
<td>Deleting the prefix unm- results in a word</td>
</tr>
<tr>
<td>20</td>
<td>NN</td>
<td>JJ</td>
<td>Has suffix -ee</td>
</tr>
</tbody>
</table>
```
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
    \[
    P(t|w) = \prod_i \frac{1}{Z(i)} \exp \left( \lambda^\top f(t_i, t_{i-1}, w, i) \right)
    \]
  - CRF
    \[
    P(t|w) = \frac{1}{Z(w)} \exp \left( \lambda^\top f(t, w) \right)
    = \frac{1}{Z(w)} \exp \left( \lambda^\top \sum_i f(t_i, t_{i-1}, w, i) \right)
    = \frac{1}{Z(w)} \prod_i \phi_i(t_i, t_{i-1})
    \]

CRFs

- Like any maxent model, derivative is:
  \[
  \frac{\partial L(\lambda)}{\partial \lambda} = \sum_k \left( f_k(t^k) - \sum_i P(t|w_k)f_k(t) \right)
  \]

- 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:

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

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

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

Forward Recurrence
Backward Recurrence

\[ \beta_t(i) = \sum_j \beta_{t+1}(j) a_{ij} b_j(o_{t+1}) \]

Fractional Transitions
EM for HMMs: Quantities

- Cache total path values:
  \[ \alpha_i(s) = P(w_0 \ldots w_i, s_i) = \sum_{s_{i-1}} P(s_i|s_{i-1})P(w_i|s_i)\alpha_{i-1}(s_{i-1}) \]
  \[ \beta_i(s) = P(w_i + 1 \ldots w_n|s_i) = \sum_{s_{i+1}} P(s_{i+1}|s_i)P(w_{i+1}|s_{i+1})\beta_{i+1}(s_{i+1}) \]
- 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

<table>
<thead>
<tr>
<th>Number of tagged sentences used for the initial model</th>
<th>0</th>
<th>100</th>
<th>2000</th>
<th>5000</th>
<th>10000</th>
<th>20000</th>
<th>all</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iter</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>77.0</td>
<td>90.0</td>
<td>95.4</td>
<td>96.2</td>
<td>96.6</td>
<td>96.9</td>
<td>97.0</td>
</tr>
<tr>
<td>1</td>
<td>80.5</td>
<td>92.6</td>
<td>95.8</td>
<td>96.3</td>
<td>96.6</td>
<td>96.7</td>
<td>96.8</td>
</tr>
<tr>
<td>2</td>
<td>81.8</td>
<td>93.0</td>
<td>95.7</td>
<td>96.1</td>
<td>96.3</td>
<td>96.4</td>
<td>96.4</td>
</tr>
<tr>
<td>3</td>
<td>83.0</td>
<td>93.1</td>
<td>95.4</td>
<td>95.8</td>
<td>96.1</td>
<td>96.2</td>
<td>96.2</td>
</tr>
<tr>
<td>4</td>
<td>84.0</td>
<td>93.0</td>
<td>95.2</td>
<td>95.5</td>
<td>95.8</td>
<td>96.0</td>
<td>96.0</td>
</tr>
<tr>
<td>5</td>
<td>84.8</td>
<td>92.9</td>
<td>95.1</td>
<td>95.4</td>
<td>95.6</td>
<td>95.8</td>
<td>95.8</td>
</tr>
<tr>
<td>6</td>
<td>85.3</td>
<td>92.8</td>
<td>94.9</td>
<td>95.2</td>
<td>95.5</td>
<td>95.6</td>
<td>95.7</td>
</tr>
<tr>
<td>7</td>
<td>85.8</td>
<td>92.8</td>
<td>94.7</td>
<td>95.1</td>
<td>95.3</td>
<td>95.5</td>
<td>95.5</td>
</tr>
<tr>
<td>8</td>
<td>86.1</td>
<td>92.7</td>
<td>94.6</td>
<td>95.0</td>
<td>95.2</td>
<td>95.4</td>
<td>95.4</td>
</tr>
<tr>
<td>9</td>
<td>86.3</td>
<td>92.6</td>
<td>94.5</td>
<td>94.9</td>
<td>95.1</td>
<td>95.3</td>
<td>95.3</td>
</tr>
<tr>
<td>10</td>
<td>86.6</td>
<td>92.6</td>
<td>94.4</td>
<td>94.8</td>
<td>95.0</td>
<td>95.2</td>
<td>95.2</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>word</th>
<th>nearest neighbors</th>
</tr>
</thead>
<tbody>
<tr>
<td>accompanied</td>
<td>submitted, banned, financed, developed, authorized, headed, canceled, awarded, barred</td>
</tr>
<tr>
<td>almost</td>
<td>virtually, merely, formally, fully, quite, officially, just, nearly, only, less</td>
</tr>
<tr>
<td>causing</td>
<td>reflecting, forcing, providing, creating, producing, becoming, carrying, particularly</td>
</tr>
<tr>
<td>classes</td>
<td>elections, courses, payments, losses, computers, performances, violations, levels, pictures</td>
</tr>
<tr>
<td>directors</td>
<td>professionals, investigations, materials, competitors, agreements, papers, transactions</td>
</tr>
<tr>
<td>goal</td>
<td>mood, roof, eye, image, tool, song, pool, scene, gap, voice</td>
</tr>
<tr>
<td>japanese</td>
<td>chinese, iran, american, western, arab, foreign, european, federal, soviet, indian</td>
</tr>
<tr>
<td>represent</td>
<td>reveal, attend, deliver, reflect, choose, contain, impose, manage, establish, retain</td>
</tr>
<tr>
<td>think</td>
<td>believe, wish, know, realize, wonder, assume, feel, say, mean, bet</td>
</tr>
<tr>
<td>york</td>
<td>angels, francisco, san, diego, broncos, las, vegas, sun, black, hawks</td>
</tr>
</tbody>
</table>

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, he, she, him, nobody, who, it, everybody, there

Dendrograms

- Pronouns: Object
- Auxiliary Verbs
- Adverbs
- WH words
- Verbs: "to be"
- Determiners
- Pronouns: Object/Possess.
- Prepositions
- Interjections
- Nouns: Proper (names)
- Adjectives: Colour, Number
- Adjectives
- Nouns
- Nouns: Proper (names)
- Verbs
- Verbs: -ing form
- Verbs
- go
- come
- sit
- stay
- stand
- start
- put
- take
- get
- cross
- give
- keep
- hold
- pick
- lock
- leave
- throw
- turn
- move
- push
- pull
- cut
- put
- pull
- put
- push
- go
- stay
- take
- over
- fall
- call
- sell
- talk
- sing
- write
- draw
- help
- eat
- hit
- break
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?

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

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)