Recap: POS Ambiguity

- Words are syntactically ambiguous:
  - Fed raises interest rates 0.5 percent

- Two sources of information:
  - Clues from the input (current word, next word, capitalization, suffixes, word shape)
  - Clues from adjacent hidden labels (connectivity)
  - What of this could HMMs capture?

- Remember: POS sequence models will be the basis of information extraction methods later

Recap: Accuracies

- Roadmap of (known / unknown) accuracies:
  - Most freq tag: ~90% / ~50%
  - Trigram HMM: ~95% / ~55%
  - Maxent P(t|w): 93.7% / 82.6%
  - TnT (HMM++): 96.2% / 86.0%
  - Maxent tagger: 96.9% / 86.9%
  - Cyclic tagger: 97.2% / 89.0%
  - Upper bound: ~98%

Most errors on unknown words

Recap: Errors

- Common errors [from Toutanova & Manning 00]

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: Importantly: -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%

Sequence-Free Tagging?

- What about looking at a word and its environment, but no sequence information?

  - Add in previous / next word shapes
  - Previous / next word shapes
  - Occurrence pattern features
  - Crude entity detection
  - Phrasal verb in sentence?
  - 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
  \[ P(t|w) = \prod_i P_M(t_i|w, t_{i-1}, t_{i-2}) \]
- Train up \( P(t|w, t_{i-1}, t_{i-2}) \) 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

- We’ve been sloppy:
  - Features: \(<w_0, t_0>\)
  - Feature templates: \(<w_0, t_0>\)
- In maxent taggers:
  - Can now add edge feature templates:
    - \(<t_{i-1}, t_i>\)
    - \(<t_i, t_{i-1}, t_{i-2}>\)
  - Also, mixed feature templates:
    - \(<t_i, w_0, t_0>\)

Decoding

- Decoding maxent taggers:
  - Just like decoding HMMs
  - Viterbi, beam search, posterior decoding
- Viterbi algorithm (HMMs):
  \[ \delta_i(s) = \arg \max_{s'} P(s|s') P(w_{i-1}|s') \delta_{i-1}(s') \]
- Viterbi algorithm (Maxent):
  \[ \delta_i(s) = \arg \max_{s'} P(s|s', w) \delta_{i-1}(s') \]

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 \(\rightarrow\) NN: DT __
    - VBD \(\rightarrow\) 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]

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: 98.6% 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
- 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

EM for HMMs: Quantities

- Remember from last time:
  \[ \alpha_t(s) = P(w_0 \ldots w_{t-1}, s_t) = \sum_{s_{t-1}} P(s_t|s_{t-1})P(w_{t-1}|s_{t-1})\alpha_{t-1}(s_{t-1}) \]
  \[ \beta_t(s) = P(w_t \ldots w_n|s_t) = \sum_{s_{t+1}} P(s_{t+1}|s_t)\beta_{t+1}(s_{t+1}) \]
- Can calculate in \(O(s^2n)\) time (why?)

EM for HMMs: Process

- From these quantities, we can re-estimate transitions:
  \[ \text{count}(s \rightarrow s') = \sum_i \alpha_i(s)P(s'|s)P(w_i|s)\beta_{i+1}(s') \]
  \[ P(w) \]
  \[ \text{And emissions:} \]
  \[ \text{count}(w, s) = \sum_i \alpha_i(s)\beta_{i+1}(s) \]
  \[ P(w) \]
- 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
    - Set \(P(w|t)\) on these examples
    - Set \(P(t|t-1,t-2)\) on these examples
  - Re-estimate with EM for n iterations
- 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>Correct tags (%) after ML on UM words</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>77.2</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>80.3</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>83.1</td>
<td>93.1</td>
<td>95.4</td>
<td>95.9</td>
<td>96.2</td>
<td>96.2</td>
<td>96.4</td>
</tr>
<tr>
<td>4</td>
<td>84.2</td>
<td>93.0</td>
<td>95.2</td>
<td>95.3</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>95.9</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>95.3</td>
<td>95.5</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>95.3</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>86.5</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>

So How to Fix It?

- Lots of progress in learning parts-of-speech
  - Distributional word clustering methods
  - Morphology-driven models
  - Contrastive estimation
  - Other ideas!

- Stay tuned…