POS Tagging
Trigram Taggers

- Trigram model: $y_1 = (\langle S\rangle, \text{NNP}), y_2 = (\text{NNP}, \text{VBZ}), ...$

- Probabilities now look like $P((\text{NNP}, \text{VBZ}) | (\langle S\rangle, \text{NNP}))$ — more context! We know the verb is occurring two words after $\langle S\rangle$

- Tradeoff between model capacity and data size — trigrams are a "sweet spot" for POS tagging

Fed raises interest rates 0.5 percent

- Normal HMM “bigram” model: $y_1 = \text{NNP}, y_2 = \text{VBZ}, ...$
HMM POS Tagging

- Penn Treebank English POS tagging (see homework): 44 tags
- Baseline: assign each word its most frequent tag: ~90% accuracy
- Trigram HMM: ~95% accuracy / 55% on words not seen in train
- TnT tagger (Brants 1998, tuned HMM): 96.2% acc / 86.0% on unks
- MaxEnt tagger (Toutanova + Manning 2000): 96.9% / 87.0%
- State-of-the-art (BiLSTM-CRFs): 97.5% / 89%+

Slide credit: Dan Klein
### Errors

<table>
<thead>
<tr>
<th>JJ</th>
<th>NN</th>
<th>NNP</th>
<th>NNPS</th>
<th>RB</th>
<th>RP</th>
<th>IN</th>
<th>VB</th>
<th>VBD</th>
<th>VBN</th>
<th>VBP</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>JJ</td>
<td>0</td>
<td>177</td>
<td>56</td>
<td>0</td>
<td>61</td>
<td>2</td>
<td>5</td>
<td>10</td>
<td>15</td>
<td>108</td>
<td>0</td>
</tr>
<tr>
<td>NN</td>
<td>244</td>
<td>0</td>
<td>103</td>
<td>0</td>
<td>12</td>
<td>1</td>
<td>1</td>
<td>29</td>
<td>5</td>
<td>6</td>
<td>19</td>
</tr>
<tr>
<td>NNP</td>
<td>107</td>
<td>106</td>
<td>0</td>
<td>132</td>
<td>5</td>
<td>0</td>
<td>7</td>
<td>5</td>
<td>1</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>NNPS</td>
<td>1</td>
<td>0</td>
<td>110</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>RB</td>
<td>72</td>
<td>21</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>16</td>
<td>138</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>RP</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>39</td>
<td>0</td>
<td>65</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>IN</td>
<td>11</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>169</td>
<td>103</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>VB</td>
<td>17</td>
<td>64</td>
<td>9</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>7</td>
<td>85</td>
</tr>
<tr>
<td>VBD</td>
<td>10</td>
<td>5</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>143</td>
<td>2</td>
</tr>
<tr>
<td>VBN</td>
<td>101</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>108</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>VBP</td>
<td>5</td>
<td>34</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>49</td>
<td>6</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>Total</td>
<td>626</td>
<td>536</td>
<td>348</td>
<td>144</td>
<td>317</td>
<td>122</td>
<td>279</td>
<td>102</td>
<td>140</td>
<td>269</td>
<td>108</td>
</tr>
</tbody>
</table>

**JJ/NN**  
official knowledge

**NN**
made up the story

**VBD**  
recently sold shares

<table>
<thead>
<tr>
<th>RP/IN</th>
<th>DT</th>
<th>NN</th>
<th>RB</th>
<th>VBD/VBN</th>
<th>NNS</th>
</tr>
</thead>
</table>

(NN NN: tax cut, art gallery, ...)

Slide credit: Dan Klein / Toutanova + Manning (2000)
Remaining Errors

- Lexicon gap (word not seen with that tag in training): 4.5% of errors
- Unknown word: 4.5%
- Could get right: 16% (many of these involve parsing!)
- Difficult linguistics: 20%

  VBD / VBP? (past or present?)
  
  *They*  *set*  *up absurd situations, detached from reality*

- Underspecified / unclear, gold standard inconsistent / wrong: 58%

  adjective or verbal participle? JJ / VBN?
  
  *a $10 million fourth-quarter charge against discontinued operations*

Manning 2011 “Part-of-Speech Tagging from 97% to 100%: Is It Time for Some Linguistics?”
Fed raises interest rates in order to ...

- Word embeddings for each word form input
- $f(x)$ doesn’t look like a bag-of-words, instead captures position-sensitive information
POS with Feedforward Networks

Botha et al. (2017): small FFNNs for NLP tasks

- Use bag-of-character bigram + trigram embeddings for each word
- Hidden layer mixes these different signals and learns feature conjunctions

Botha et al. (2017)
POS with Feedforward Networks

- Works well on a range of languages
- Better than a RNN-based approach (Gillick et al., 2016)

<table>
<thead>
<tr>
<th>Lang.</th>
<th>L.R.</th>
<th>Mom.</th>
<th>$\gamma$</th>
<th>Steps</th>
<th>Acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>bg</td>
<td>0.1</td>
<td>0.8</td>
<td>128k</td>
<td>210k</td>
<td>97.76</td>
</tr>
<tr>
<td>cs</td>
<td>0.05</td>
<td>0.9</td>
<td>32k</td>
<td>420k</td>
<td>98.06</td>
</tr>
<tr>
<td>da</td>
<td>0.05</td>
<td>0.9</td>
<td>16k</td>
<td>240k</td>
<td>95.33</td>
</tr>
<tr>
<td>en</td>
<td>0.05</td>
<td>0.8</td>
<td>8k</td>
<td>300k</td>
<td>93.06</td>
</tr>
<tr>
<td>fi</td>
<td>0.05</td>
<td>0.9</td>
<td>16k</td>
<td>390k</td>
<td>94.66</td>
</tr>
<tr>
<td>fr</td>
<td>0.08</td>
<td>0.9</td>
<td>128k</td>
<td>120k</td>
<td>95.28</td>
</tr>
<tr>
<td>de</td>
<td>0.08</td>
<td>0.9</td>
<td>16k</td>
<td>90k</td>
<td>92.13</td>
</tr>
<tr>
<td>el</td>
<td>0.08</td>
<td>0.9</td>
<td>16k</td>
<td>60k</td>
<td>97.42</td>
</tr>
<tr>
<td>id</td>
<td>0.08</td>
<td>0.9</td>
<td>8k</td>
<td>690k</td>
<td>92.15</td>
</tr>
<tr>
<td>it</td>
<td>0.05</td>
<td>0.9</td>
<td>64k</td>
<td>210k</td>
<td>97.42</td>
</tr>
<tr>
<td>fa</td>
<td>0.1</td>
<td>0.8</td>
<td>8k</td>
<td>510k</td>
<td>96.19</td>
</tr>
<tr>
<td>es</td>
<td>0.08</td>
<td>0.9</td>
<td>8k</td>
<td>60k</td>
<td>94.79</td>
</tr>
<tr>
<td>sv</td>
<td>0.1</td>
<td>0.8</td>
<td>16k</td>
<td>300k</td>
<td>95.76</td>
</tr>
</tbody>
</table>

Botha et al. (2017)
CRFs and NER
Barack Obama will travel to Hangzhou today for the G20 meeting.

- Frame as a sequence problem with a BIO tagset: begin, inside, outside
- Why might an HMM not do so well here?
  - Lots of O’s, so tags aren’t as informative about context
  - Want to use context features (to Hangzhou => Hangzhou is a LOC)
- Conditional random fields (CRFs) can help solve these problems
HMMs

- Big advantage: transitions, scoring pairs of adjacent y’s

- Big downside: not able to incorporate useful word context information

- Solution: switch from generative to discriminative model (conditional random fields) so we can condition on the entire input.

- Conditional random fields: logistic regression + features on pairs of y’s
Tagging with Logistic Regression

- Logistic regression over each tag individually: “different features” approach to features for a single tag

\[ P(y_i = y|x, i) = \frac{\exp(w^T f(y, i, x))}{\sum_{y' \in Y} \exp(w^T f(y', i, x))} \]

- Over all tags:

\[ P(y = \tilde{y}|x) = \prod_{i=1}^{n} P(y_i = \tilde{y}_i|x, i) = \frac{1}{Z} \exp \left( \sum_{i=1}^{n} w^T f(\tilde{y}_i, i, x) \right) \]

- Score of a prediction: sum of weights dot features over each individual predicted tag (this is a simple CRF but not the general form)

- Set Z equal to the product of denominators; we’ll discuss this in a few slides
Adding Structure

\[ P(y = \tilde{y} | x) = \frac{1}{Z} \exp \left( \sum_{i=1}^{n} w^\top f_{e}(\tilde{y}_i, i, x) \right) \]

- We want to be able to learn that some tags don’t follow other tags — want to have features on tag pairs

\[ P(y = \tilde{y} | x) = \frac{1}{Z} \exp \left( \sum_{i=1}^{n} w^\top f_{e}(\tilde{y}_i, i, x) + \sum_{i=1}^{n} w^\top f_{t}(\tilde{y}_i, \tilde{y}_{i+1}, i, x) \right) \]

- Score: sum of weights dot \( f_{e} \) features over each predicted tag (“emissions”) plus sum of weights dot \( f_{t} \) features over tag pairs (“transitions”)

- This is a sequential CRF
Barack Obama will travel to Hangzhou today for the G20 meeting.

Transitions: \( f_t(O, B-LOC, i = 5, x) = \text{Indicator}[O \rightarrow B-LOC] \)

Emissions: \( f_e(B-LOC, i = 6, x) = \text{Indicator}[B-LOC \& \text{Curr word prefix} = H] \)
\( \text{Indicator}[B-LOC \& \text{Prev word} = \text{to}] \)

- We couldn’t use a “previous word” feature in the HMM at all!
Features for NER

- Current word features (can use in HMM)
  - Capitalization
  - Word shape
  - Prefixes/suffixes
  - Lexical indicators
- Context features (can’t use in HMM!)
  - Words before/after
  - Tags before/after
- Word clusters/embeddings
- Gazetteers

According to the New York Times...

Apple released a new version...

Leicestershire

Boston
Example

- CRFs assign a score to every possible tag sequence over a sentence.

  **Example 1**
  
  \[
  \begin{align*}
  &B-PER \quad I-PER \quad O \quad O \\
  &\text{Barack Obama will travel}
  \\
  &\text{feats} = f_e(B-PER, i=1, x) + f_e(I-PER, i=2, x) + f_e(O, i=3, x) + f_e(O, i=4, x) \\
  &\quad + f_t(B-PER, I-PER, i=1, x) + f_t(I-PER, O, i=2, x) + f_t(O, O, i=3, x)
  \end{align*}
  \]

  **Example 2**
  
  \[
  \begin{align*}
  &B-PER \quad B-PER \quad O \quad O \\
  &\text{Barack Obama will travel}
  \\
  &\text{feats} = f_e(B-PER, i=1, x) + f_e(B-PER, i=2, x) + f_e(O, i=3, x) + f_e(O, i=4, x) \\
  &\quad + f_t(B-PER, B-PER, i=1, x) + f_t(B-PER, O, i=2, x) + f_t(O, O, i=3, x)
  \end{align*}
  \]

- *Obama* can start a new named entity (*emission feats* look okay), but we’re not likely to have two PER entities in a row (*transition feats*)
Conditional Random Fields

- **HMMs:** \( P(y, x) = P(y_1)P(x_1|y_1)P(y_2|y_1)P(x_2|y_2) \ldots \)

- **CRFs:** discriminative models with the following globally-normalized form:

\[
P(y = \tilde{y} | x) = \frac{1}{Z} \exp \left( \sum_{i=1}^{n} w^T f_e(\tilde{y}_i, i, x) + \sum_{i=1}^{n} w^T f_t(\tilde{y}_i, \tilde{y}_{i+1}, i, x) \right)
\]

normalizer \( Z \): must make this a probability distribution over all possible seqs

\[
Z = \sum_{y' \in \mathcal{Y}^n} \exp \left( \sum_{i=1}^{n} w^T f_e(y'_i, i, x) + \sum_{i=1}^{n} w^T f_t(y'_i, y'_{i+1}, i, x) \right)
\]

- **CRFs in general:** replace weights dot features with so-called “potential functions” over \( y \)’s
Inference and Learning

\[ P(\mathbf{y} = \tilde{\mathbf{y}} | \mathbf{x}) = \frac{1}{Z} \exp \left( \sum_{i=1}^{n} \mathbf{w}^\top \mathbf{f}_e(\tilde{y}_i, i, \mathbf{x}) + \sum_{i=1}^{n} \mathbf{w}^\top \mathbf{f}_t(\tilde{y}_i, \tilde{y}_{i+1}, i, \mathbf{x}) \right) \]

- **Inference:** Can use the Viterbi algorithm to find the highest scoring path if we replace HMM log probs with “scores” from weights dot features (initial distribution is removed)

  \[
  \log P(x_i|y_i) \rightarrow \mathbf{w}^\top \mathbf{f}_e(y_i, i, \mathbf{x})
  \]

  \[
  \log P(y_i|y_{i-1}) \rightarrow \mathbf{w}^\top \mathbf{f}_t(y_{i-1}, y_i, i, \mathbf{x})
  \]

- **Learning:** requires running *forward-backward* (like Viterbi but with summing instead of maxing over \( y \)'s) to compute \( Z \), then doing some tricky math to compute gradients [outside scope of the course/not on midterm]
Takeaways

- CRFs provide a way to build structured feature-based models: logistic regression over structured objects like sequences

- Inference and learning can still be done efficiently but require dynamic programming

- CRFs don’t have to be linear models; can use scores derived from neural networks ("neural CRFs")