Feature-based Tagging

Instructor: Yoav Artzi

Slides adapted from Dan Klein, Luke Zettlemoyer, Chris Manning, and Dan Jurafsky
Re-visit $P(y \mid x)$

- Reality check:
  - What if we drop the sequence?
    - Re-visit $P(y \mid x)$?
  - Most frequent tag:
    - 90.3% with a bad unknown word model
    - 93.7% with a good one
  - Can we do better?
What about better features?

• Looking at a word and its environment
  – 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

• Uses lots of features: > 200K
Some Numbers

• Rough accuracies:
  – Most freq tag: ~90% / ~50%
  – Trigram HMM: ~95% / ~55%
  – TnT (Brants, 2000): 96.7% / 85.5%
  – MaxEnt $P(y \mid x)$

• What does this tell us about sequence models?
• How do we add more features to our sequence models?
MEMM Taggers

One step up: also condition on previous tags:

\[ p(s_1 \ldots s_m | x_1 \ldots x_m) = \prod_{i=1}^{m} p(s_i | s_1 \ldots s_{i-1}, x_1 \ldots x_m) \]

\[ = \prod_{i=1}^{m} p(s_i | s_{i-1}, x_1 \ldots x_m) \]

- Training:
  - Train \( p(s_i | s_{i-1}, x_1 \ldots x_m) \) as a discrete log-linear (MaxEnt) model
- Scoring:

\[ p(s_i | s_{i-1}, x_1 \ldots x_m) = \frac{\exp \left( w \cdot \phi(x_1 \ldots x_m, i, s_{i-1}, s_i) \right)}{\sum_{s'} \exp \left( w \cdot \phi(x_1 \ldots x_m, i, s_{i-1}, s') \right)} \]

- This is referred to as an MEMM tagger [Ratnaparkhi 96]
HMM vs. MEMM

• HMM models joint distribution:

\[ p(x_1 \ldots x_n, y_1 \ldots y_n) = q(STOP|y_n) \prod_{i=1}^{n} q(y_i|y_{i-1}) e(x_i|y_i) \]

• MEMM models conditioned distribution:

\[ p(s_1 \ldots s_m|x_1 \ldots x_m) = \prod_{i=1}^{m} p(s_i|s_1 \ldots s_{i-1}, x_1 \ldots x_m) \]
Decoding MEMM Taggers

• Scoring:

\[ p(s_i|s_{i-1}, x_1 \ldots x_m) = \text{__________________________} \]

• Beam search is effective – why?
• Guarantees? Optimal?
• Can we do better?
The State Lattice / Trellis

- Fed raises interest rates

$e(\text{Fed}|\text{N})$
$e(\text{raises}|\text{V})$
$e(\text{interest}|\text{V})$
$e(\text{rates}|\text{J})$
$e(\text{STOP}|\text{V})$

START
Fed
raises
interest
rates
STOP

^ N V J D
^ N V J D
^ N V J D
^ N V J D
^ N V J D
^ N V J D
The MEMM State Lattice / Trellis

START

Fed raises interest rates

STOP
Decoding MEMM Taggers

- Decoding MaxEnt taggers:
  - Just like decoding HMMs
  - Viterbi, beam search, posterior decoding

- Viterbi algorithm (HMMs):
  - Define $\pi(i, s_i)$ to be the max score of a sequence of length $i$ ending in tag $s_i$

$$
\pi(i, s_i) = \max_{s_{i-1}} e(x_i | s_i) q(s_i | s_{i-1}) \pi(i - 1, s_{i-1})
$$

- Viterbi algorithm (Maxent):
  - Can use same algorithm for MEMMs, just need to redefine $\pi(i, s_i)$!

$$
\pi(i, s_i) = \max_{s_{i-1}} p(s_i | s_{i-1}, x_1 \ldots x_m) \pi(i - 1, s_{i-1})
$$
Some Numbers

• Rough accuracies:
  – Most freq tag: ~90% / ~50%
  – Trigram HMM: ~95% / ~55%
  – TnT (Brants, 2000): 96.7% / 85.5%
  – MaxEnt P(y | x) 96.8% / 86.8%
  – MEMM tagger 1:
Feature Development

Common errors:

<table>
<thead>
<tr>
<th></th>
<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>
<td>488</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>
<td>525</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>
<td>427</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>
<td>142</td>
</tr>
<tr>
<td>RB</td>
<td>72</td>
<td>21</td>
<td>7</td>
<td>0</td>
<td>16</td>
<td>138</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>295</td>
</tr>
<tr>
<td>RP</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>39</td>
<td>65</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>104</td>
</tr>
<tr>
<td>IN</td>
<td>11</td>
<td>0</td>
<td>1</td>
<td>169</td>
<td>103</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>323</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>
<td>189</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>
<td>166</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>
<td>221</td>
</tr>
<tr>
<td>VBP</td>
<td>5</td>
<td>34</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>49</td>
<td>6</td>
<td>3</td>
<td>0</td>
<td>104</td>
<td></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>
<td>3651</td>
</tr>
</tbody>
</table>

NN/JJ NN
official knowledge

VBD RP/IN DT NN
made up the story

RB VBD/VBN NNS
recently sold shares

[Toutanova and Manning 2000]
Some Numbers

• Rough accuracies:
  – Most freq tag: ~90% / ~50%
  – Trigram HMM: ~95% / ~55%
  – TnT (Brants, 2000): 96.7% / 85.5%
  – MaxEnt P(y | x) 96.8% / 86.8%
  – MEMM tagger 1: 96.6% / 85.5%
  – MEMM tagger 2:

[Toutanova and Manning 2000]
Locally Normalized Models

• So far:
  – Probabilities are product of *locally normalized* probabilities
  – Is this bad?

<table>
<thead>
<tr>
<th>from \ to</th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.4</td>
<td>0.4</td>
<td>0.2</td>
<td>0.4</td>
</tr>
<tr>
<td>B</td>
<td>0.0</td>
<td>1.0</td>
<td>0.0</td>
</tr>
<tr>
<td>C</td>
<td>0.6</td>
<td>0.2</td>
<td>0.2</td>
</tr>
</tbody>
</table>

B → B transitions are likely to take over even if rarely seen!
Locally Normalized Models

• So far:
  – Probabilities are product of locally normalized probabilities
  – Is this bad?

• Label bias
  – MEMM 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
Global Discriminative Taggers

• Newer, higher-powered discriminative sequence models
  – CRFs (also Perceptrons)
  – Do not decompose training into independent local regions
  – Can be deathly slow to train – require repeated inference on training set
Linear Models: Perceptron

• The perceptron algorithm
  – Iteratively processes the data, reacting to training errors
  – Can be thought of as trying to drive down training error

• The (online structured) perceptron algorithm:
  – Start with zero weights
  – Visit training instances \((x_i, y_i)\) one by one
    • Make a prediction
      \[
      y^* = \arg \max_y w \cdot \phi(x_i, y)
      \]
    • If correct \((y^* = y_i)\):
      – no change, goto next example!
    • If wrong:
      – adjust weights: \[
      w = w + \phi(x_i, y_i) - \phi(x_i, y^*)
      \]

• Challenge: How to compute argmax efficiently?
Decoding

• **Linear Perceptron** \[ s^* = \arg \max_s \mathbf{w} \cdot \Phi(x, s) \cdot \theta \]
  
  – Features must be local, for \( x=x_1...x_m \), and \( s=s_1...s_m \)

\[
\Phi(x, s) = \sum_{j=1}^{m} \phi(x, j, s_{j-1}, s_j)
\]
The MEMM State Lattice / Trellis

START

Fed raises interest rates

STOP
The Perceptron State Lattice / Trellis

START
^ Fed raises interest rates STOP
^ N V V J V
V N N N N N N
N N V V V V
J J J J J J
D D D D D D D
$ $ $ $ $ $ $
Decoding

• Linear Perceptron: $s^* = \arg \max_s w \cdot \Phi(x, s) \cdot \theta$
  
  Features must be local, for $x=x_1...x_m$, and $s=s_1...s_m$

  $\Phi(x, s) = \sum_{j=1}^{m} \phi(x, j, s_{j-1}, s_j)$

  - Define $\pi(i,s_i)$ to be the max score of a sequence of length $i$ ending in tag $s_i$

  $$\pi(i, s_i) = \max_{s_{i-1}} w \cdot \phi(x, i, s_{i-i}, s_i) + \pi(i - 1, s_{i-1})$$

• Viterbi algorithm (HMMs):

  $$\pi(i, s_i) = \max_{s_{i-1}} e(x_i | s_i) q(s_i | s_{i-1}) \pi(i - 1, s_{i-1})$$

• Viterbi algorithm (Maxent):

  $$\pi(i, s_i) = \max_{s_{i-1}} p(s_i | s_{i-1}, x_1 \ldots x_m) \pi(i - 1, s_{i-1})$$
Some Numbers

• Rough accuracies:
  – Most freq tag: ~90% / ~50%
  – Trigram HMM: ~95% / ~55%
  – TnT (Brants, 2000): 96.7% / 85.5%
  – MaxEnt $P(y \mid x)$: 96.8% / 86.8%
  – MEMM tagger 1: 96.6% / 85.5%
  – MEMM tagger 2: 96.8% / 86.9%
  – Perceptron: [Collins 2002]
Conditional Random Fields (CRFs)

- What did we lose with the Perceptron?
  - No probabilities
  - Let’s try again with a probabilistic model
CRFs

- Maximum entropy (logistic regression)

Sentence: $x = x_1 \ldots x_m$

Tag Sequence: $y = s_1 \ldots s_m$

- **Learning**: maximize the (log) conditional likelihood of training data

$$p(y|x; w) = \frac{1}{Z(x)} \exp(w \cdot \phi(x, y))$$

$$L(w) = \frac{1}{n} \sum_{i=1}^{n} \left( \phi_j(x_i, y_i) - \sum_y p(y|x_i; w) \phi_j(x_i, y) \right) - \lambda w_j$$

- **Computational Challenges**?
  - Most likely tag sequence, normalization constant, gradient

[Lafferty et al. 2001]
Decoding

• CRFs
  – Features must be local, for $x=x_1\ldots x_m$, and $s=s_1\ldots s_m$

\[
p(y|x; w) = \frac{\exp (w \cdot \Phi(x, y))}{\sum_{y'} \exp (w \cdot \Phi(x, y'))}
\]

\[
\Phi(x, y) = \sum_{j=1}^{m} \phi(x, j, y_{j-1}, y_j)
\]

\[
\arg \max_y \frac{\exp (w \cdot \Phi(x, y))}{\sum_{y'} \exp (w \cdot \Phi(x, y'))} = \arg \max_y \exp (w \cdot \Phi(x, y))
\]

\[
= \arg \max_y w \cdot \Phi(x, y)
\]

• Same as Linear Perceptron!

\[
\pi(i, y_i) = \max_{y_{i-1}} \phi(x, i, y_{i-1}, y_i) + \pi(i - 1, y_{i-1})
\]
CRFs: Computing Normalization

\[
p(y|x; w) = \frac{\exp (w \cdot \Phi(x, y))}{\sum_{y'} \exp (w \cdot \Phi(x, y'))} \quad \Phi(x, y) = \sum_{j=1}^{m} \phi(x, j, y_{j-1}, y_j)
\]

\[
\sum_{y'} \exp (w \cdot \Phi(x, y')) = \sum_{y'} \exp \left( \sum_j w \cdot \phi(x, j, y_{j-1}, y_j) \right)
\]

\[
= \sum \prod_{y', j} \exp (w \cdot \phi(x, j, y_{j-1}, y_j))
\]

Define \( \text{norm}(i,s_i) \) to sum of scores for sequences ending in position \( i \)

\[
\text{norm}(i, y_i) = \sum_{y_{i-1}} \exp (w \cdot \phi(x, i, y_{i-1}, y_i)) \text{norm}(i - 1, y_{i-1})
\]

- **Forward Algorithm!** Remember HMM case:

\[
\alpha(i, y_i) = \sum_{y_{i-1}} e(x_i|y_i)q(y_i|y_{i-1})\alpha(i - 1, y_{i-1})
\]

  - Could also use backward?
CRFs: Computing Gradient

\[ p(y|x; w) = \frac{\exp(w \cdot \Phi(x, y))}{\sum_{y'} \exp(w \cdot \Phi(x, y'))} \]

\[ \Phi(x, y) = \sum_{k=1}^{m} \phi(x, k, y_{k-1}, y_k) \]

\[ \frac{\partial}{\partial w_j} L(w) = \sum_{i=1}^{n} \left( \Phi_j(x_i, y_i) - \sum_{y} p(y|x_i; w) \Phi_j(x_i, y) \right) - \lambda w_j \]

\[ \sum_{y} p(y|x_i; w) \Phi_j(x_i, y) = \sum_{y} p(y|x_i; w) \sum_{k=1}^{m} \phi_j(x_i, k, y_{k-1}, y_k) \]

\[ = \sum_{k=1}^{m} \sum_{a,b} \sum_{y:y_{k-1}=a,y_k=b} p(y|x_i; w) \phi_j(x_i, k, y_{k-1}, y_k) \]

- Need forward and backward messages

See CRF notes for full details!
Some Numbers

• Rough accuracies:
  – Most freq tag: ~90% / ~50%
  – Trigram HMM: ~95% / ~55%
  – TnT (Brants, 2000): 96.7% / 85.5%
  – MaxEnt $P(y \mid x)$: 96.8% / 86.8%
  – MEMM tagger 1: 96.6% / 85.5%
  – MEMM tagger 2: 96.8% / 86.9%
  – Perceptron: 97.1%
  – CRF++:

[Sun 2014]
Cyclic Network

- Train two MEMMs, multiple together to score
- And be very careful
  - Tune regularization
  - Try lots of different features
  - See paper for full details

(a) Left-to-Right CMM
(b) Right-to-Left CMM
(c) Bidirectional Dependency Network

[Toutanova et al. 2003]
Some Numbers

- Rough accuracies:
  - Most freq tag: ~90% / ~50%
  - Trigram HMM: ~95% / ~55%
  - TnT (Brants, 2000): 96.7% / 85.5%
  - MaxEnt $P(y \mid x)$ 96.8% / 86.8%
  - MEMM tagger 1: 96.6% / 85.5%
  - MEMM tagger 2: 96.8% / 86.9%
  - Perceptron: 97.1%
  - CRF++: 97.3%
  - Cyclic tagger:
  - Upper bound: ~98%

[Toutanova et al. 2003]
Summary

• For tagging, the change from generative to discriminative model does not by itself result in great improvement.
• One profits from models for specifying dependence on overlapping features of the observation such as spelling, suffix analysis, etc.
• MEMMs allow integration of rich features of the observations.
• This additional power (of the MEMM, CRF, Perceptron models) has been shown to result in improvements in accuracy.
• The higher accuracy of discriminative models comes at the price of much slower training.
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)