Sequence Prediction and Part-of-speech Tagging

Instructor: Yoav Artzi

Slides adapted from Dan Klein, Dan Jurafsky, Chris Manning, Michael Collins, Luke Zettlemoyer, Yejin Choi, and Slav Petrov
Overview

• POS Tagging: the problem
• Hidden Markov Models (HMM)
  – Supervised Learning
  – Inference
    • The Viterbi algorithm
• Feature-rich models
  – Maximum-entropy Markov Models
  – Perceptron
  – Conditional Random Fields
Parts of Speech

Open class (lexical) words

Nouns
- Proper: IBM, Italy
- Common: cat / cats, snow

Verbs
- Main: see, registered

Adjectives
- old, older, oldest

Closed class (functional)

Determiners: the, some

Conjunctions: and, or

Pronouns: he, its

Modals: can, had

Prepositions: to, with

Particles: off, up

Interjections: Ow, Eh

Numbers: 122,312, one

Adverbs: slowly

… more
POS Tagging

- Words often have more than one POS: *back*
  - The *back* door = JJ
  - On my *back* = NN
  - Win the voters *back* = RB
  - Promised to *back* the bill = VB

- The POS tagging problem is to determine the POS tag for a particular instance of a word.
POS Tagging

• Input: Plays well with others
• Ambiguity: NNS/VBZ UH/JJ/NN/RB IN NNS
• Output: Plays/VBZ well/RB with/IN others/NNS
• Uses:
  – Text-to-speech (how do we pronounce “lead” ?)
  – Can write regular expressions like (Det) Adj* N+ over the output for phrases, etc.
  – As input to a full parser (e.g., to create dependency trees)
  – If you know the tag, you can back off to it in other tasks
Penn TreeBank Tagset

• Possible tags: 45
• Tagging guidelines: 36 pages
• Newswire text
Penn TreeBank Tagset

- How accurate are taggers? (Tag accuracy)
  - About >97% currently
  - But baseline is already 90%
    - Baseline is performance of simplest possible method
      - Tag every word with its most frequent tag
      - Tag unknown words as nouns
    - Partly easy because
      - Many words are unambiguous
      - You get points for them (the, a, etc.) and for punctuation marks!
  - Upperbound: probably 2% annotation errors
Hard Cases are Hard

- Mrs/NNP Shaefer/NNP never/RB got/VBD around/RP to/TO joining/VBG

- All/DT we/PRP gotta/VBN do/VB is/VBZ go/VB around/IN the/DT corner/NN

- Chateau/NNP Petrus/NNP costs/VCZ around/RB 250/CD
How Difficult is POS Tagging?

• About 11% of the word types in the Brown corpus are ambiguous with regard to part of speech
• But they tend to be very common words. E.g., *that*
  – I know *that* he is honest = IN
  – Yes, *that* play was nice = DT
  – You can’t go *that* far = RB
• 40% of the word tokens are ambiguous
The Tagset

• Wait, do we really need all these tags?
• What about other languages?
  – Each language has its own tagset
## Tagsets in Different Languages

<table>
<thead>
<tr>
<th>Language</th>
<th>Source</th>
<th># Tags</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arabic</td>
<td>PADT/CoNLL07 (Hajič et al., 2004)</td>
<td>21</td>
</tr>
<tr>
<td>Basque</td>
<td>Basque3LB/CoNLL07 (Aduriz et al., 2003)</td>
<td>64</td>
</tr>
<tr>
<td>Bulgarian</td>
<td>BTB/CoNLL06 (Simov et al., 2002)</td>
<td>54</td>
</tr>
<tr>
<td>Catalan</td>
<td>CESS-ECE/CoNLL07 (Martí et al., 2007)</td>
<td>54</td>
</tr>
<tr>
<td>Chinese</td>
<td>Penn ChineseTreebank 6.0 (Palmer et al., 2007)</td>
<td>34</td>
</tr>
<tr>
<td>Chinese</td>
<td>Sinica/CoNLL07 (Chen et al., 2003)</td>
<td>294</td>
</tr>
<tr>
<td>Czech</td>
<td>PDT/CoNLL07 (Böhmová et al., 2003)</td>
<td>63</td>
</tr>
<tr>
<td>Danish</td>
<td>DDT/CoNLL06 (Kromann et al., 2003)</td>
<td>25</td>
</tr>
<tr>
<td>Dutch</td>
<td>Alpino/CoNLL06 (Van der Beek et al., 2002)</td>
<td>12</td>
</tr>
<tr>
<td>English</td>
<td>PennTreebank (Marcus et al., 1993)</td>
<td>45</td>
</tr>
<tr>
<td>French</td>
<td>FrenchTreebank (Abeillé et al., 2003)</td>
<td>30</td>
</tr>
<tr>
<td>German</td>
<td>Tiger/CoNLL06 (Brants et al., 2002)</td>
<td>54</td>
</tr>
<tr>
<td>German</td>
<td>Negra (Skut et al., 1997)</td>
<td>54</td>
</tr>
<tr>
<td>Greek</td>
<td>GDT/CoNLL07 (Prokopidis et al., 2005)</td>
<td>38</td>
</tr>
<tr>
<td>Hungarian</td>
<td>Szeged/CoNLL07 (Csendes et al., 2005)</td>
<td>43</td>
</tr>
<tr>
<td>Italian</td>
<td>ISST/CoNLL07 (Montemagni et al., 2003)</td>
<td>28</td>
</tr>
<tr>
<td>Japanese</td>
<td>Verbmbobil/CoNLL06 (Kawata and Bartels, 2000)</td>
<td>80</td>
</tr>
<tr>
<td>Japanese</td>
<td>Kyoto4.0 (Kurohashi and Nagao, 1997)</td>
<td>42</td>
</tr>
<tr>
<td>Korean</td>
<td>Sejong (<a href="http://www.sejong.or.kr">http://www.sejong.or.kr</a>)</td>
<td>187</td>
</tr>
<tr>
<td>Portuguese</td>
<td>Floresta Sintá(c)tica/CoNLL06 (Afonso et al., 2002)</td>
<td>22</td>
</tr>
<tr>
<td>Russian</td>
<td>SynTagRus-RNC (Boguslavsky et al., 2002)</td>
<td>11</td>
</tr>
<tr>
<td>Slovene</td>
<td>SDT/CoNLL06 (Džeroski et al., 2006)</td>
<td>29</td>
</tr>
<tr>
<td>Spanish</td>
<td>Ancora-Cast3LB/CoNLL06 (Civit and Martí, 2004)</td>
<td>47</td>
</tr>
<tr>
<td>Swedish</td>
<td>Talbanken05/CoNLL06 (Nivre et al., 2006)</td>
<td>41</td>
</tr>
<tr>
<td>Turkish</td>
<td>METU-Sabanci/CoNLL07 (Oflazer et al., 2003)</td>
<td>31</td>
</tr>
</tbody>
</table>
The Tagset

• Wait, do we really need all these tags?
• What about other languages?
  – Each language has its own tagset
    • But why is this bad?
    • Differences in downstream tasks
    • Harder to do language transfer
Alternative: The Universal Tagset

• 12 tags:
  – NOUN, VERB, ADJ, ADV, PRON, DET, ADP, NUM, CONJ, PRT, ‘.’, and X.

• Deterministic conversion from tagsets in 22 languages.

• Better unsupervised parsing results

• Was used to transfer parsers

[ Petrov et al. 2012]
Sources of Information

• What are the main sources of information for POS tagging?
  – Knowledge of neighboring words
    • Bill saw that man yesterday
    • NNP VB(D) DT NN NN
    • VB NN IN VB NN
  – Knowledge of word probabilities
    • *man* is rarely used as a verb….  

• The latter proves the most useful, but the former also helps
Word-level Features

• Can do surprisingly well just looking at a word by itself:
  – Word the: the → DT
  – Lowercased words: importantly → RB
  – Prefixes unfathomable: un- → JJ
  – Suffixes Importantly: -ly → RB
  – Capitalization Meridian: CAP → NNP
  – Word shapes 35-year: d-x → JJ
Consider the problem of jointly modeling a pair of strings
– E.g.: part of speech tagging

<table>
<thead>
<tr>
<th>DT</th>
<th>NNP</th>
<th>NN</th>
<th>VBD</th>
<th>VBN</th>
<th>RP</th>
<th>NN</th>
<th>NNS</th>
</tr>
</thead>
<tbody>
<tr>
<td>The</td>
<td>Georgia branch</td>
<td>had</td>
<td>taken</td>
<td>on</td>
<td>loan</td>
<td>commitments</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DT</th>
<th>NN</th>
<th>IN</th>
<th>NN</th>
<th>VBD</th>
<th>NNS</th>
<th>VBD</th>
</tr>
</thead>
<tbody>
<tr>
<td>The</td>
<td>average of</td>
<td>interbank</td>
<td>offered</td>
<td>rates</td>
<td>plummeted</td>
<td></td>
</tr>
</tbody>
</table>

Q: How do we map each word in the input sentence onto the appropriate label?
A: We can learn a joint distribution:

\[ p(x_1 \ldots x_n, y_1 \ldots y_n) \]

And then compute the most likely assignment:

\[ \arg \max_{y_1 \ldots y_n} p(x_1 \ldots x_n, y_1 \ldots y_n) \]
Classic Solution: HMMs

We want a model of sequences \( y \) and observations \( x \)

\[
p(x_1 \ldots x_n, y_1 \ldots y_n) =
\]

where \( y_0 = START \) and we call \( q(y_i | y_{i-1}) \) the transition distribution and \( e(x_i | y_i) \) the emission (or observation) distribution.
Model Assumptions

- Tag/state sequence is generated by a Markov model
- Words are chosen independently, conditioned only on the tag/state
- These are totally broken assumptions for POS: why?

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

The Georgia branch had taken on loan commitments …

DT  NNP  NN  VBD  VBN  RP  NN  NNS

• HMM Model:
  – States \( Y = \)
  – Observations \( X = \)
  – Transition dist’n \( q(y_i | y_{i-1}) \) models
  – Emission dist’n \( e(x_i | y_i) \) models
HMM for POS Tagging

The Georgia branch had taken on loan commitments …

- States $Y = \{\text{DT, NNP, NN, ...} \}$ are the POS tags
- Observations $X = V$ are words
- Transition dist’n $q(y_i|y_{i-1})$ models the tag sequences
- Emission dist’n $e(x_i|y_i)$ models words given their POS
HMM Inference and Learning

- Learning
  - Maximum likelihood: transitions $q$ and emissions $e$
  
  $$p(x_1 \ldots x_n, y_1 \ldots y_n) = q(STOP|y_n) \prod_{i=1}^{n} q(y_i|y_{i-1}) e(x_i|y_i)$$

- Inference
  - Viterbi
  
  $$y^* = \arg \max_{y_1 \ldots y_n} p(x_1 \ldots x_n, y_1 \ldots y_n)$$

  - Forward backward
  
  $$p(x_1 \ldots x_n, y_i) = \sum_{y_1 \ldots y_{i-1}} \sum_{y_{i+1} \ldots y_n} p(x_1 \ldots x_n, y_1 \ldots y_n)$$
Learning: Maximum Likelihood

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

- Maximum likelihood methods for estimating transitions \( q \) and emissions \( e \)

\[ q_{ML}(y_i|y_{i-1}) = \frac{c(y_{i-1}, y_i)}{c(y_{i-1})} \quad e_{ML}(x|y) = \frac{c(y, x)}{c(y)} \]

- Will these estimates be high quality?
  - Which is likely to be more sparse, \( q \) or \( e \)?
- Smoothing?
Learning: Low Frequency Words

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

- Typically, for transitions:
  - Linear Interpolation
  
  \[ q(y_i|y_{i-1}) = \lambda_1 q_{ML}(y_i|y_{i-1}) + \lambda_2 q_{ML}(y_i) \]

- However, other approaches used for emissions
  - **Step 1:** Split the vocabulary
    - Frequent words: appear more than \( M \) (often 5) times
    - Low frequency: everything else
  - **Step 2:** Map each low frequency word to one of a small, finite set of possibilities
    - For example, based on prefixes, suffixes, etc.
  - **Step 3:** Learn model for this new space of possible word sequences
Another Example: Chunking

- Goal: Segment text into spans with certain properties
- For example, named entities: PER, ORG, and LOC

Germany’s representative to the European Union’s veterinary committee Werner Zwingman said on Wednesday consumers should…

[Germany]_{LOC} ’s representative to the [European Union]_{ORG} ’s veterinary committee [Werner Zwingman]_{PER} said on Wednesday consumers should…

How is this a sequence tagging problem?
Named Entity Recognition

Germany’s representative to the European Union’s veterinary committee Werner Zwingman said on Wednesday consumers should…

• HMM Model:
  – States $Y = \{\text{NA}, \text{BL}, \text{CL}, \text{BO}, \text{CO}, \text{BP}, \text{CP}\}$ represent beginnings (BL, BO, BP) and continuations (CL, CO, CP) of chunks, as well as other words (NA)
  – Observations $X = V$ are words
  – Transition dist’n $q(y_i|y_{i-1})$ models the tag sequences
  – Emission dist’n $e(x_i|y_i)$ models words given their type
Low Frequency Words: An Example

- **Named Entity Recognition [Bickel et. al, 1999]**
  - Used the following word classes for infrequent words:

<table>
<thead>
<tr>
<th>Word class</th>
<th>Example</th>
<th>Intuition</th>
</tr>
</thead>
<tbody>
<tr>
<td>twoDigitNum</td>
<td>90</td>
<td>Two digit year</td>
</tr>
<tr>
<td>fourDigitNum</td>
<td>1990</td>
<td>Four digit year</td>
</tr>
<tr>
<td>containsDigitAndAlpha</td>
<td>A8956-67</td>
<td>Product code</td>
</tr>
<tr>
<td>containsDigitAndDash</td>
<td>09-96</td>
<td>Date</td>
</tr>
<tr>
<td>containsDigitAndSlash</td>
<td>11/9/89</td>
<td>Date</td>
</tr>
<tr>
<td>containsDigitAndComma</td>
<td>23,000.00</td>
<td>Monetary amount</td>
</tr>
<tr>
<td>containsDigitAndPeriod</td>
<td>1.00</td>
<td>Monetary amount, percentage</td>
</tr>
<tr>
<td>othernum</td>
<td>456789</td>
<td>Other number</td>
</tr>
<tr>
<td>allCaps</td>
<td>BBN</td>
<td>Organization</td>
</tr>
<tr>
<td>capPeriod</td>
<td>M.</td>
<td>Person name initial</td>
</tr>
<tr>
<td>firstWord</td>
<td>first word of sentence</td>
<td>no useful capitalization information</td>
</tr>
<tr>
<td>initCap</td>
<td>Sally</td>
<td>Capitalized word</td>
</tr>
<tr>
<td>lowercase</td>
<td>can</td>
<td>Uncapitalized word</td>
</tr>
<tr>
<td>other</td>
<td>,</td>
<td>Punctuation marks, all other words</td>
</tr>
</tbody>
</table>
Low Frequency Words: An Example

Profits/NA soared/NA at/NA Boeing/SO Co./CO ,/NA easily/NA topping/NA forecasts/NA on/NA Wall/SL Street/CL ,/NA as/NA their/NA CEO/NA Alan/SP Mulally/CP announced/NA first/NA quarter/NA results/NA ./NA

- NA = No entity
- SO = Start Organization
- CO = Continue Organization
- SL = Start Location
- CL = Continue Location
- ...
Low Frequency Words: An Example

Profits/NA soared/NA at/NA Boeing/SO Co./CO ,/NA easily/NA topping/NA forecasts/NA on/NA Wall/SL Street/CL ,/NA as/NA their/NA CEO/NA Alan/SP Mulally/CP announced/NA first/NA quarter/NA results/NA ./NA

firstword/NA soared/NA at/NA initCap/SC Co./CC ,/NA easily/NA lowercase/NA forecasts/NA on/NA initCap/SL Street/CL ,/NA as/NA their/NA CEO/NA Alan/SP initCap/CP announced/NA first/NA quarter/NA results/NA ./NA

- NA = No entity
- SO = Start Organization
- CO = Continue Organization
- SL = Start Location
- CL = Continue Location
- ...
HMM Inference and Learning

• Learning
  – Maximum likelihood: transitions $q$ and emissions $e$
    $$p(x_1 \ldots x_n, y_1 \ldots y_n) = q(STOP|y_n) \prod_{i=1}^{n} q(y_i|y_{i-1})e(x_i|y_i)$$

• Inference
  – Viterbi
    $$y^* = \arg\max_{y_1 \ldots y_n} p(x_1 \ldots x_n, y_1 \ldots y_n)$$
  – Forward backward
    $$p(x_1 \ldots x_n, y_i) = \sum_{y_1 \ldots y_{i-1}} \sum_{y_{i+1} \ldots y_n} p(x_1 \ldots x_n, y_1 \ldots y_n)$$
Inference (Decoding)

• Problem: find the most likely (Viterbi) sequence under the model

\[ y^* = \arg \max_{y_1 \ldots y_n} p(x_1 \ldots x_n, y_1 \ldots y_n) \]

• Given model parameters, we can score any sequence pair

NNP VBZ NN NNS CD NN .
Fed raises interest rates 0.5 percent .

\[ q(\text{NNP} | \quad ) e(\text{Fed} | \text{NNP}) q(\text{VBZ} | \text{NNP}) e(\text{raises} | \text{VBZ}) q(\text{NN} | \text{VBZ}) \ldots \]

• In principle, we’re done – list all possible tag sequences, score each one, pick the best one (the Viterbi state sequence)

NNP VBZ NN NNS CD NN .  \( \log p(x, y) = -23 \)
NNP NNS NN NNS CD NN .  \( \log(x, y) = -29 \)
NNP VBZ VB NNS CD NN .  \( \log p(x, y) = -27 \)

Any issue?
Finding the Best Trajectory

• Too many trajectories (state sequences) to list
• Option 1: Beam Search
  – A beam is a set of partial hypotheses
  – Start with just the single empty trajectory
  – At each derivation step:
    • Consider all continuations of previous hypotheses
    • Discard most, keep top k

• Beam search often works OK in practice, but …
  • … sometimes you want the optimal answer
  • … and there’s usually a better option than naïve beams
The State Lattice / Trellis

The diagram shows a state lattice with transitions labeled by events. The states are labeled with a capital letter: N, V, J, D, and $.

The events include:
- e(Fed|N)
- e(raises|V)
- e(interest|V)
- e(rates|J)
- e(STOP|V)

The diagram starts with an arrow labeled q(N/V) from the start state to the state N, and other transitions are marked with probabilities and events.

The events correspond to the sequence: Fed raises interest rates STOP
Scoring a Sequence

\[ y^* = \arg \max_{y_1 \ldots y_n} p(x_1 \ldots x_n, y_1 \ldots y_n) \]

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

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

\[ \pi(i, y_i) = \max_{y_1 \ldots y_{i-1}} p(x_1 \ldots x_i, y_1 \ldots y_i) \]

\[ = \max_{y_{i-1}} e(x_i|y_i)q(y_i|y_{i-1}) \max_{y_1 \ldots y_{i-2}} p(x_1 \ldots x_{i-1}, y_1 \ldots y_{i-1}) \]

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

- We can now design an efficient algorithm.
  - How?
The Viterbi Algorithm

Dynamic program for computing (for all $i$)
$$\pi(i, y_i) = \max_{y_1 \ldots y_{i-1}} p(x_1 \ldots x_i, y_1 \ldots y_i)$$

Iterative computation:
$$\pi(0, y_0) = \begin{cases} 1 & \text{if } y_0 == START \\ 0 & \text{otherwise} \end{cases}$$

For $i = 1 \ldots n$:

// Store score
$$\pi(i, y_i) = \max_{y_{i-1}} e(x_i|y_i) q(y_i|y_{i-1}) \pi(i - 1, y_{i-1})$$

// Store back-pointer
$$bp(i, y_i) = \arg \max_{y_{i-1}} e(x_i|y_i) q(y_i|y_{i-1}) \pi(i - 1, y_{i-1})$$
The State Lattice / Trellis

Tie breaking: Prefer first

START
Fed raises interest STOP

<table>
<thead>
<tr>
<th>from \ to</th>
<th>^</th>
<th>N</th>
<th>V</th>
<th>$</th>
</tr>
</thead>
<tbody>
<tr>
<td>^</td>
<td>0.0</td>
<td>0.6</td>
<td>0.4</td>
<td>0.0</td>
</tr>
<tr>
<td>N</td>
<td>0.0</td>
<td>0.4</td>
<td>0.2</td>
<td>0.4</td>
</tr>
<tr>
<td>V</td>
<td>0.0</td>
<td>0.6</td>
<td>0.1</td>
<td>0.3</td>
</tr>
<tr>
<td>$</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>emissions</th>
<th>START</th>
<th>Fed</th>
<th>raises</th>
<th>interest</th>
<th>STOP</th>
</tr>
</thead>
<tbody>
<tr>
<td>^</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>N</td>
<td>0.0</td>
<td>0.45</td>
<td>0.1</td>
<td>0.45</td>
<td>0.0</td>
</tr>
<tr>
<td>V</td>
<td>0.0</td>
<td>0.0</td>
<td>0.7</td>
<td>0.4</td>
<td>0.0</td>
</tr>
<tr>
<td>$</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
</tr>
</tbody>
</table>
The State Lattice / Trellis

Tie breaking:
Prefer first

<table>
<thead>
<tr>
<th>from \ to</th>
<th>^</th>
<th>N</th>
<th>V</th>
<th>$</th>
</tr>
</thead>
<tbody>
<tr>
<td>^</td>
<td>0.0</td>
<td>0.6</td>
<td>0.4</td>
<td>0.0</td>
</tr>
<tr>
<td>N</td>
<td>0.0</td>
<td>0.4</td>
<td>0.2</td>
<td>0.4</td>
</tr>
<tr>
<td>V</td>
<td>0.0</td>
<td>0.6</td>
<td>0.1</td>
<td>0.3</td>
</tr>
<tr>
<td>$</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>emissions</th>
<th>START</th>
<th>Fed</th>
<th>raises</th>
<th>interest</th>
<th>STOP</th>
</tr>
</thead>
<tbody>
<tr>
<td>^</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>N</td>
<td>0.0</td>
<td>0.45</td>
<td>0.1</td>
<td>0.45</td>
<td>0.0</td>
</tr>
<tr>
<td>V</td>
<td>0.0</td>
<td>0.0</td>
<td>0.7</td>
<td>0.4</td>
<td>0.0</td>
</tr>
<tr>
<td>$</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
</tr>
</tbody>
</table>
The Viterbi Algorithm: Runtime

- In term of sentence length $n$?
  - Linear
- In term of number of states $|\mathcal{K}|$?
  - Polynomial

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

- Specifically:

\[
O(n|\mathcal{K}|) \text{ entries in } \pi(i, y_i)
\]
\[
O(|\mathcal{K}|) \text{ time to compute each } \pi(i, y_i)
\]

- Total runtime: \( O(n|\mathcal{K}|^2) \)

Q: Is this a practical algorithm?
A: depends on $|\mathcal{K}|$....
## Tagsets in Different Languages

<table>
<thead>
<tr>
<th>Language</th>
<th>Source</th>
<th># Tags</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arabic</td>
<td>PADT/CoNLL07 (Hajič et al., 2004)</td>
<td>21</td>
</tr>
<tr>
<td>Basque</td>
<td>Basque3LB/CoNLL07 (Aduriz et al., 2003)</td>
<td>64</td>
</tr>
<tr>
<td>Bulgarian</td>
<td>BTB/CoNLL06 (Simov et al., 2002)</td>
<td>54</td>
</tr>
<tr>
<td>Catalan</td>
<td>CESS-ECE/CoNLL07 (Martí et al., 2007)</td>
<td>54</td>
</tr>
<tr>
<td>Chinese</td>
<td>Penn ChineseTreebank 6.0 (Palmer et al., 2007)</td>
<td>24</td>
</tr>
<tr>
<td>Chinese</td>
<td>Sinica/CoNLL07 (Chen et al., 2003)</td>
<td>294</td>
</tr>
<tr>
<td>Czech</td>
<td>PDT/CoNLL07 (Böhmová et al., 2003)</td>
<td>63</td>
</tr>
<tr>
<td>Danish</td>
<td>DDT/CoNLL06 (Kromann et al., 2003)</td>
<td>25</td>
</tr>
<tr>
<td>Dutch</td>
<td>Alpino/CoNLL06 (Van der Beek et al., 2002)</td>
<td>12</td>
</tr>
<tr>
<td>English</td>
<td>PennTreebank (Marcus et al., 1993)</td>
<td>45</td>
</tr>
<tr>
<td>French</td>
<td>FrenchTreebank (Abeillé et al., 2003)</td>
<td>30</td>
</tr>
<tr>
<td>German</td>
<td>Tiger/CoNLL06 (Brants et al., 2002)</td>
<td>54</td>
</tr>
<tr>
<td>German</td>
<td>Negra (Skut et al., 1997)</td>
<td>54</td>
</tr>
<tr>
<td>Greek</td>
<td>GDT/CoNLL07 (Prokopidis et al., 2005)</td>
<td>38</td>
</tr>
<tr>
<td>Hungarian</td>
<td>Szeged/CoNLL07 (Csendes et al., 2005)</td>
<td>43</td>
</tr>
<tr>
<td>Italian</td>
<td>ISST/CoNLL07 (Montemagni et al., 2003)</td>
<td>28</td>
</tr>
<tr>
<td>Japanese</td>
<td>Verb mobil/CoNLL06 (Kawata and Bartels, 2000)</td>
<td>80</td>
</tr>
<tr>
<td>Japanese</td>
<td>Kyoto4.0 (Kurohashi and Nagao, 1997)</td>
<td>42</td>
</tr>
<tr>
<td>Korean</td>
<td>Sejong (<a href="http://www.sejong.or.kr">http://www.sejong.or.kr</a>)</td>
<td>187</td>
</tr>
<tr>
<td>Portuguese</td>
<td>Floresta Sintá(ct)ica/CoNLL06 (Afonso et al., 2002)</td>
<td>22</td>
</tr>
<tr>
<td>Russian</td>
<td>SynTagRus-RNC (Boguslavsky et al., 2002)</td>
<td>11</td>
</tr>
<tr>
<td>Slovene</td>
<td>SDT/CoNLL06 (Džeroski et al., 2006)</td>
<td>20</td>
</tr>
<tr>
<td>Spanish</td>
<td>Ancora-Cast3LB/CoNLL06 (Civit and Martí, 2004)</td>
<td>47</td>
</tr>
<tr>
<td>Swedish</td>
<td>Talbanken05/CoNLL06 (Nivre et al., 2006)</td>
<td>41</td>
</tr>
<tr>
<td>Turkish</td>
<td>METU-Sabanci/CoNLL07 (Oflazer et al., 2003)</td>
<td>31</td>
</tr>
</tbody>
</table>

294² = 86436

45² = 2045

11² = 121

[Source: Petrov et al. 2012]
HMM Inference and Learning

• Learning
  – Maximum likelihood: transitions $q$ and emissions $e$
    \[
p(x_1 \ldots x_n, y_1 \ldots y_n) = q(\text{STOP}|y_n) \prod_{i=1}^{n} q(y_i|y_{i-1}) e(x_i|y_i)
    \]

• Inference
  – Viterbi
    \[
y^* = \arg \max_{y_1 \ldots y_n} p(x_1 \ldots x_n, y_1 \ldots y_n)
    \]
  – Forward backward
    \[
p(x_1 \ldots x_n, y_i) = \sum_{y_1 \ldots y_{i-1}} \sum_{y_{i+1} \ldots y_n} p(x_1 \ldots x_n, y_1 \ldots y_n)
    \]
What about n-gram Taggers?

- States encode what is relevant about the past
- Transitions $P(s_i | s_{i-1})$ encode well-formed tag sequences
  - In a bigram tagger, states = tags
    - $s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow \ldots \rightarrow s_n$
    - $x_1 \rightarrow x_2 \rightarrow \ldots \rightarrow x_n$
  - In a trigram tagger, states = tag pairs
    - $s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow \ldots \rightarrow s_n$
    - $x_1 \rightarrow x_2 \rightarrow \ldots \rightarrow x_n$
The State Lattice / Trellis

Not all edges are allowed

\[ \text{START} \]

\[ \wedge \]

\[ \text{Fed} \]

\[ N \]

\[ \wedge \]

\[ \text{raises} \]

\[ D \]

\[ \wedge \]

\[ \text{interest} \]

\[ V \]

\[ \wedge \]

\[ \text{...} \]

\[ \wedge \]

\[ \text{...} \]

\[ \wedge \]

\[ \text{...} \]

\[ \wedge \]

\[ \text{...} \]

\[ \wedge \]

\[ \text{...} \]

\[ \wedge \]

\[ \text{...} \]

\[ \wedge \]

\[ \text{...} \]
## Tagsets in Different Languages

<table>
<thead>
<tr>
<th>Language</th>
<th>Source</th>
<th># Tags</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arabic</td>
<td>PADT/CoNLL07 (Hajič et al., 2004)</td>
<td>21</td>
</tr>
<tr>
<td>Basque</td>
<td>Basque3LB/CoNLL07 (Aduriz et al., 2003)</td>
<td>64</td>
</tr>
<tr>
<td>Bulgarian</td>
<td>BTB/CoNLL06 (Simov et al., 2002)</td>
<td>54</td>
</tr>
<tr>
<td>Catalan</td>
<td>CESS-ECE/CoNLL07 (Martí et al., 2007)</td>
<td>54</td>
</tr>
<tr>
<td>Chinese</td>
<td>Penn ChineseTreebank 6.0 (Palmer et al., 2007)</td>
<td>31</td>
</tr>
<tr>
<td>Chinese</td>
<td>Sinica/CoNLL07 (Chen et al., 2003)</td>
<td>294</td>
</tr>
<tr>
<td>Czech</td>
<td>PDT/CoNLL07 (Böhmová et al., 2003)</td>
<td>63</td>
</tr>
<tr>
<td>Danish</td>
<td>DDT/CoNLL06 (Krommann et al., 2003)</td>
<td>25</td>
</tr>
<tr>
<td>Dutch</td>
<td>Alpino/CoNLL06 (Van der Beek et al., 2002)</td>
<td>12</td>
</tr>
<tr>
<td>English</td>
<td>PennTreebank (Marcus et al., 1993)</td>
<td>45</td>
</tr>
<tr>
<td>French</td>
<td>FrenchTreebank (Abeillé et al., 2003)</td>
<td>30</td>
</tr>
<tr>
<td>German</td>
<td>Tiger/CoNLL06 (Brants et al., 2002)</td>
<td>54</td>
</tr>
<tr>
<td>German</td>
<td>Negra (Skut et al., 1997)</td>
<td>54</td>
</tr>
<tr>
<td>Greek</td>
<td>GDT/CoNLL07 (Prokopidis et al., 2005)</td>
<td>38</td>
</tr>
<tr>
<td>Hungarian</td>
<td>Szeged/CoNLL07 (Csendes et al., 2005)</td>
<td>43</td>
</tr>
<tr>
<td>Italian</td>
<td>ISST/CoNLL07 (Montemagni et al., 2003)</td>
<td>28</td>
</tr>
<tr>
<td>Japanese</td>
<td>Verbmobil/CoNLL06 (Kawata and Bartels, 2000)</td>
<td>80</td>
</tr>
<tr>
<td>Japanese</td>
<td>Kyoto4.0 (Kurohashi and Nagao, 1997)</td>
<td>42</td>
</tr>
<tr>
<td>Korean</td>
<td>Sejong (<a href="http://www.sejong.or.kr">http://www.sejong.or.kr</a>)</td>
<td>187</td>
</tr>
<tr>
<td>Portuguese</td>
<td>Floresta Sintá(c)tica/CoNLL06 (Afonso et al., 2002)</td>
<td>22</td>
</tr>
<tr>
<td>Russian</td>
<td>SynTagRus-RNC (Boguslavsky et al., 2002)</td>
<td>11</td>
</tr>
<tr>
<td>Slovene</td>
<td>SDT/CoNLL06 (Džeroski et al., 2006)</td>
<td>30</td>
</tr>
<tr>
<td>Spanish</td>
<td>Ancora-Cast3LB/CoNLL06 (Civit and Martí, 2004)</td>
<td>47</td>
</tr>
<tr>
<td>Swedish</td>
<td>Talbanken05/CoNLL06 (Nivre et al., 2006)</td>
<td>41</td>
</tr>
<tr>
<td>Turkish</td>
<td>METU-Sabanci/CoNLL07 (Oflazer et al., 2003)</td>
<td>31</td>
</tr>
</tbody>
</table>

\[
294^2 = 86436 \\
294^4 = 7471182096 \\
45^2 = 2045 \\
45^4 = 4100625 \\
11^2 = 121 \\
11^4 = 14641
\]
Some Numbers

• Rough accuracies:
  – Most freq tag:
  – Trigram HMM:
  – TnT (Brants, 2000):
    • A carefully smoothed trigram tagger
    • Suffix trees for emissions

– Upper bound: ~98%
Re-visit $P(x \mid y)$

• Reality check:
  – What if we drop the sequence?
    • Use only $P(x \mid y)$
  – Most frequent tag:
    • 90.3% with a so-so unknown word model
  – 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 | x)

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

  – Upper bound: ~98%
MEMM Taggers

One step up: also condition on previous tags:

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

\[ = \prod_{i=1}^{n} p(y_i|y_{i-1}, x_1 \ldots x_n) \]

• Training:
  – Train \( p(y_i|y_{i-1}, x_1 \ldots x_n) \) as a discrete log-linear (MaxEnt) model
• Scoring:

\[ p(y_i|y_{i-1}, x_1 \ldots x_n) = \frac{e^{w \cdot \phi(x_1 \ldots x_n, i, y_{i-1}, y_i)}}{\sum_{y'} e^{w \cdot \phi(x_1 \ldots x_n, i, y_{i-1}, y')}} \]

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

- HMM models joint distribution:

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

- MEMM models conditioned distribution:

\[
p(y_1 \ldots y_n|x_1 \ldots x_n) = \prod_{i=1}^{n} p(y_i|y_1 \ldots y_{i-1}, x_1 \ldots x_n)
\]
Decoding MEMM Taggers

• Scoring:

\[ p(y_i|y_{i-1}, x_1 \ldots x_n) = \frac{e^{w \cdot \phi(x_1 \ldots x_n, i, y_{i-1}, y_i)}}{\sum_{y'} e^{w \cdot \phi(x_1 \ldots x_n, i, y_{i-1}, y')}} \]

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

```
The State Lattice / Trellis

START       Fed           raises       interest       rates       STOP
^                N               V                 V                    J               V

q(N|^) e(Fed|N) q(V|N) e(Fed|V) e(V|N) e(raises|V) e(interest|V) e(rates|J) e(STOP|V)

```

The State Lattice / Trellis

```
The State Lattice / Trellis

START       Fed           raises       interest       rates       STOP
^                N               V                 V                    J               V

q(N|^) e(Fed|N) q(V|N) e(Fed|V) e(V|N) e(raises|V) e(interest|V) e(rates|J) e(STOP|V)

`````
The MEMM State Lattice / Trellis

START       Fed           raises       interest       rates       STOP
^             N               V              V               J               V

q(N|^, X)
q(V|N, X)
q(V|V, X)
q(V|V, X)
q(V|V, X)
q(V|V, X)
q(V|V, X)
Decoding MEMM Taggers

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

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

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

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

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

- Rough accuracies:
  - Most freq tag: ~90% / ~50%
  - Trigram HMM: ~95% / ~55%
  - TnT (Brants, 2000): 96.7% / 85.5%
  - MaxEnt P(y | x): 93.7% / 82.6%
  - MEMM tagger 1:
  - Upper bound: ~98%
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>0</td>
<td>0</td>
<td>16</td>
<td>1</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>39</td>
<td>0</td>
<td>0</td>
<td>65</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>0</td>
<td>2</td>
<td>49</td>
<td>6</td>
<td>3</td>
<td>0</td>
<td>104</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

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) 93.7% / 82.6%
  – MEMM tagger 1: 96.7% / 84.5%
  – MEMM tagger 2:

  – Upper bound: ~98%

[Toutanova and Manning 2000]
Locally Normalized Models

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

• Label bias
  – States with fewer transitions are likely to be preferred because normalization is local
  – Extreme case: What happens if there is only one outgoing arc? Does it matter what the observation is?
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>A</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 observed!
Global Discriminative Taggers

• Discriminative sequence models
  – CRFs (also Perceptrons)
  – Do not decompose training into independent local regions
  – Can be very slow* to train – require repeated inference on training set

* Relatively slow. NN models are much slower.
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 \(\arg\max\) efficiently?
Decoding

• **Linear Perceptron** \( Y^* = \arg \max_Y w \cdot \phi(X, Y) \)
  - Features must be local, for \( X = x_1 \ldots x_n \), and \( Y = y_1 \ldots y_m \)

\[
\phi(X, Y) = \sum_{j=1}^{n} \phi(X, j, y_{j-1}, y_j)
\]
The MEMM State Lattice / Trellis

START       Fed           raises       interest         rates         STOP
START       Fed           raises       interest       rates       STOP
\(^\)                N               V              V                   J               V
Decoding

• **Linear Perceptron** \( Y^* = \arg \max_Y w \cdot \phi(X, Y) \)
  
  – Features must be local, for \( X = x_1 \ldots x_n \), and \( Y = y_1 \ldots y_n \)
  
  \[ \phi(X, Y) = \sum_{j=1}^n \phi(X, j, y_{j-1}, y_j) \]
  
  – Define \( \pi(i, y_i) \) to be the max score of a sequence of length \( i \) ending in tag \( y_i \)

\[ \pi(i, y_i) = \max_{y_{i-1}} w \cdot \phi(X, i, y_{i-1}, y_i) + \pi(i - 1, y_{i-1}) \]

• **Viterbi algorithm (HMMs):**

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

• **Viterbi algorithm (Maxent):**

\[ \pi(i, y_i) = \max_{y_{i-1}} p(y_i | y_{i-1}, x_1 \ldots x_m) \pi(i - 1, y_{i-1}) \]
Some Numbers

• Rough accuracies:
  – Most freq tag: ~90% / ~50%
  – Trigram HMM: ~95% / ~55%
  – TnT (Brants, 2000): 96.7% / 85.5%
  – MaxEnt P(y | x): 93.7% / 82.6%
  – MEMM tagger 1: 96.7% / 84.5%
  – MEMM tagger 2: 96.8% / 86.9%
  – Perceptron:
    – Upper bound: ~98%
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: $X = x_1 \ldots x_n$

Tag Sequence: $Y = y_1 \ldots y_n$

- **Learning**: maximize the (log) conditional likelihood of training data $\{(X^{(i)}, Y^{(i)})\}_{i=1}^m$

\[
\frac{\partial}{\partial w_j} L(w) = \sum_{i=1}^m \left( \phi_j(X^{(i)}, Y^{(i)}) - \sum_Y p(Y \mid X^{(i)}; w) \phi_j(X^{(i)}, Y) \right) - \lambda w_j
\]

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

[Lafferty et al. 2001]
Decoding

- CRFs
  - Features must be local, for $x = x_1 \ldots x_n$, and $y = y_1 \ldots y_n$

$$p(Y \mid X; w) = \frac{\exp(w \cdot \phi(X, Y))}{\sum_{Y'} \exp(w \cdot \phi(X, Y'))}$$

$$\phi(X, Y) = \sum_{j=1}^{n} \phi(X, j, y_{j-1}, y_j)$$

$$\arg \max_Y \frac{\exp(w \cdot \phi(X, Y))}{\sum_{Y'} \exp(w \cdot \phi(X, Y'))} = \arg \max_Y \exp(w \cdot \phi(X, Y))$$

$$= \arg \max_Y w \cdot \phi(X, Y)$$

- Looks familiar?
- Same as linear Perceptron!

$$\pi(i, y_i) = \max_{y_{i-1}} \phi(x, i, y_{i-1}, y_i) + \pi(i - 1, y_{i-1})$$

$$Y^* = \arg \max_Y p(Y \mid X; w)$$
CRFs: Computing Normalization

\[
p(Y \mid X; w) = \frac{\exp(w \cdot \phi(X, Y))}{\sum_{Y'} \exp(w \cdot \phi(X, Y'))} \quad \phi(X, Y) = \sum_{j=1}^{n} \phi(X, j, y_{j-1}, y_j)
\]

\[
\sum_{Y'} \exp(w \cdot \phi(X, Y')) = \sum_{Y'} \exp \left( \sum_{j=1}^{n} w \cdot \phi(X, j, y_{j-1}, y_j) \right)
\]

\[
= \sum_{Y'} \prod_{j=1}^{n} \exp(w \cdot \phi(X, j, y_{j-1}, y_j))
\]

Define \(\text{norm}(i, y_i)\) 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:

\[
\pi(i, y_i) = \max_{y_{i-1}} e(x_i \mid y_i) q(y_i \mid y_{i-1}) \pi(i - 1, y_{i-1})
\]
CRFs: Computing Gradient

\[ p(Y \mid X; w) = \frac{\exp(w \cdot \phi(X, Y))}{\sum_{Y'} \exp(w \cdot \phi(X, Y'))} \]

\[ \phi(X, Y) = \sum_{j=1}^{n} \phi(X, j, y_{j-1}, y_{j}) \]

\[ \frac{\partial}{\partial w_j} L(w) = \sum_{I=1}^{m} \left( \phi_j(X^{(i)}, Y^{(i)}) - \sum_{Y} p(Y \mid X^{(i)}; w) \phi_j(X^{(i)}, Y) \right) - \lambda w_j \]

\[ \sum_{Y} p(Y \mid X^{(i)}; w) \phi_j(X^{(i)}, Y) = \sum_{Y} p(Y \mid X^{(i)}; w) \sum_{k=1}^{n} \phi_j(X^{(i)}, k, y_{k-1}, y_{k}) \]

\[ = \sum_{k=1}^{n} \sum_{a,b} \sum_{y_{k-1}=a, y_{k}=b} p(Y \mid X^{(i)}; w) \phi_j(X^{(i)}, k, y_{k-1}, y_{k}) \]

- Can compute with the Forward Backward algorithm

See 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 $P(y \mid x)$: 93.7% / 82.6%
  - MEMM tagger 1: 96.7% / 84.5%
  - MEMM tagger 2: 96.8% / 86.9%
  - Perceptron: 97.1%
  - CRF++:
    - Upper bound: ~98%
Cyclic Network

- Train two MEMMs, combine scores
- 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

[Ottanova 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 | x) 93.7% / 82.6%
  - MEMM tagger 1: 96.7% / 84.5%
  - MEMM tagger 2: 96.8% / 86.9%
  - Perceptron: 97.1%
  - CRF++: 97.3%
  - Cyclic tagger:
  - Upper bound: ~98%

[Toutanova et al. 2003]
Summary

• Generative vs. discriminative
• Probabilistic or not
  – Probabilities are great for upstream tasks
  – But: label bias, global normalization, etc.
• Structured or not
  – Independent predictions are effective, but global structure matters
  – But: need to balance global vs. local for tractability
• Model expressivity
  – Higher n-grams are better
  – But: cost
Summary

- For tagging, the change from generative to discriminative model does not by itself result in great improvement.
- But: profit from models by 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)