Natural Language Processing

Feature-rich models

Based on slides from Michael Collins and Yoav Artzi
Motivation
Back to language modeling

• Goal: estimate a distribution \( p(w_i \mid w_1, w_2, \ldots, w_{i-1}) \)

• Example (Chomsky, 1950):

Third, the notion “grammatical in English” cannot be identified in any way with the notion “high order statistical approximation to English”. It is fair to assume that neither sentence (1) nor (2) (nor indeed any part of these sentences) has ever occurred in an English discourse. Hence in any statistical _______
Trigram models

Third, the notion “grammatical in English” cannot be identified in any way with the notion “high order statistical approximation to English”. It is fair to assume that neither sentence (1) nor (2) (nor indeed any part of these sentences) has ever occurred in an English discourse. Hence in any statistical _______

\[
q_{LI}(\text{model} \mid w_{i-2}, w_{i-1}) = \lambda_1 \times q(\text{model} \mid w_{i-2} = \text{any}, w_{i-1} = \text{statistical}) \\
+ \lambda_2 \times q(\text{model} \mid w_{i-1} = \text{statistical}) \\
+ \lambda_3 \times q(\text{model}) \\
\lambda_i \geq 0, \quad \lambda_1 + \lambda_2 + \lambda_3 = 1
\]

\[
q(y \mid x) = \frac{c(x, y)}{c(x)}
\]
Problems

• We use unigrams, bigrams and trigrams information

• But there is a lot of other information ("features"):

\[ q(\text{model} \mid w_{i-2} = \text{any}) \]
\[ q(\text{model} \mid w_{i-1} \text{ is an adjective}) \]
\[ q(\text{model} \mid w_{i-1} \text{ ends in "ical"}) \]
\[ q(\text{model} \mid \text{author is Chomsky}) \]
\[ q(\text{model} \mid "\text{model}" \text{ does not occur in } w_1, \ldots, w_{i-1}) \]
\[ \vdots \]
Naive approach

\[ \lambda_1 q(\ldots) + \lambda_2 q(\ldots) \cdots + \lambda_F q(\ldots) \]

- Estimate a ML model for each information piece
- Interpolate and learn the coefficient
  - Is this scalable?
Profits soared at Boeing Co., easily topping forecasts on Wall Street, as their CEO Alan Mulally announced first quarter results.
Tagging

• We learned to estimate $p(w, t)$

• We want to define a model that directly estimates $p(t \mid w)$

• Why?

\[
p(t_1, \ldots, t_n \mid w_1, \ldots, w_n, t_{-1}, t_0) = p(t_1 \mid t_{-1}, t_0, w_1, \ldots, w_n) \\
\times p(t_2 \mid t_{-1}, t_0, t_1, w_1, \ldots, w_n) \\
\vdots \\
\times \ldots p(t_n \mid t_{-1}, \ldots, t_{n-1}, w_1, \ldots, w_n) \\
= \prod_{i=1}^{n} p(t_i \mid t_{-1}, \ldots, t_{i-1}, w_1, \ldots, w_n)
\]
Tagging

• We want a model

\[ p(t_i \mid t_1, \ldots, t_{i-1}, w_1, \ldots, w_n) \]

Hispaniola/NNP quickly/RB became/VB an/DT important/JJ base/?? from which Spain expanded its empire into the rest of the Western Hemisphere.

• This is a multi-class problem with classes NN, NNS, Vt, Vi, IN, DT, …
Information pieces

- $q(\text{NN} \mid w_i=\text{base})$: local feature
- $q(\text{NN} \mid t_{i-1} \text{ is JJ})$: context feature
- $q(\text{NN} \mid w_i \text{ ends in “e”})$
- $q(\text{NN} \mid w_i \text{ ends in “se”})$
- $q(\text{NN} \mid w_{i-1} \text{ is “important”})$
- $q(\text{NN} \mid w_{i+1} \text{ is “from”})$
Features in log-linear models
Definition

• **Input:**
  
  • domain $X$
    
    • LM: document prefixes + index of next position $i$
    
    • What is it in tagging?

  • finite label set
    
    • words in vocabulary (LM)
    
    • tag set in tagging

• **Output** a model $p(y \mid x)$
Features

• A feature is a function from \((x, y)\) to a real value:

\[
f_k(x, y) \in \mathbb{R}
\]

\[
f_k(x, y) \in \{0, 1\}
\]

• A feature vector holds \(m\) features

\[
f(x, y) = \langle f_1(x, y), f_2(x, y), \ldots, f_m(x, y) \rangle
\]
Third, the notion “grammatical in English” cannot be identified in any way with the notion “high order statistical approximation to English”. It is fair to assume that neither sentence (1) nor (2) (nor indeed any part of these sentences) has ever occurred in an English discourse. Hence in any statistical ______

\[
f_1(x, y) = \begin{cases} 
1 & \text{if } y = \text{model} \\
0 & \text{otherwise}
\end{cases}
\]

\[
f_2(x, y) = \begin{cases} 
1 & \text{if } y = \text{model}, \ x_{i-1} = \text{statistical} \\
0 & \text{otherwise}
\end{cases}
\]

\[
f_3(x, y) = \begin{cases} 
1 & \text{if } y = \text{model}, \ x_{i-1} = \text{statistical}, \ x_{i-2} = \text{any} \\
0 & \text{otherwise}
\end{cases}
\]
More features

\[ f_4(x, y) = \begin{cases} 
1 & \text{if } y = \text{model}, \ x_{i-2} = \text{any} \\
0 & \text{otherwise} 
\end{cases} \]

\[ f_5(x, y) = \begin{cases} 
1 & \text{if } y = \text{model}, \ x_{i-1} \text{ is an adjective} \\
0 & \text{otherwise} 
\end{cases} \]

\[ f_6(x, y) = \begin{cases} 
1 & \text{if } y = \text{model}, \ x_{i-1} \text{ ends with “ical”} \\
0 & \text{otherwise} 
\end{cases} \]

\[ f_7(x, y) = \begin{cases} 
1 & \text{if } y = \text{model}, \text{ author is Chomsky} \\
0 & \text{otherwise} 
\end{cases} \]
Feature templates

• If we have a feature over pairs or triples of objects then we would have a feature for every possible combination.

• For example, for every triple of words $u$, $v$, $w$ that appears in the training data we have a feature:

$$f_{N(u,v,w)}(x, y) = \begin{cases} 
1 & \text{if } y = w, x_{i-1} = v, x_{i-2} = u \\
0 & \text{otherwise}
\end{cases}$$

• $N(u, v, w)$ is a function from the word triple to an non-negative integer.
Tagging features

\[ x = \langle t_1, \ldots, t_{i-1}, w_1, \ldots, w_n, i \rangle \]
\[ y \in S \quad (\text{NN}, \text{NNS}, \ldots) \]
\[ f(x, y) \in \mathbb{R}^m \]
\[ f_1(x, y) = \begin{cases} 
1 & \text{if } w_i = \text{base and } y = \text{Vt} \\
0 & \text{otherwise} 
\end{cases} \]
\[ f_1(x, y) = \begin{cases} 
1 & \text{if } w_i \text{ ends in “ing” and } y = \text{VBG} \\
0 & \text{otherwise} 
\end{cases} \]

- Use feature templates for pairing all word/tag pairs or all suffix/tag pairs
Feature representation

- Feature vectors are sparse - mostly zeros

- How many features are on in a LM with trigram, bigram and unigram feature templates?

- Represent with hash maps (or hash sets if all features are binary):

  \[
  f(\langle\ldots, DT, JJ, Hispaniola, \ldots, 6\rangle, Vt) = \{121, 1003, 21554\}
  
  f(\langle\ldots, DT, JJ, Hispaniola, \ldots, 6\rangle, JJ) = \{3, 115, 2051\}
  \]

- Efficient in memory and dot-product computation (\(f'\) is a dense representation and \(f''\) is a sparse one)

  \[
  f(x, y)^\top \theta = \sum_k f_k'(x, y) \cdot \theta_k = \sum_{k \in f''_k(x, y)} \theta_k
  \]
Class 6
Profits soared at Boeing Co., easily topping forecasts on Wall Street, as their CEO Alan Mulally announced first quarter results.
HMMs (Reminder)

\[
p(x_1, \ldots, x_n, y_1 \ldots, y_n, \text{STOP}) = \\
q(\text{STOP} \mid y_{n-1}, y_n) \times \prod_{i=1}^{n} q(y_i \mid y_{i-2}, y_{i-1}) \cdot e(x_i \mid y_i)
\]

\[
y_0 = y_{-1} = *
\]
Learning (Reminder)

\[
q(y_i \mid y_{i-2}, y_{i-1}) = \lambda_1 \cdot \frac{c(y_{i-2}, y_{i-1}, y_i)}{c(y_{i-2}, y_{i-1})} + 
\lambda_2 \cdot \frac{c(y_{i-1}, y_i)}{c(y_{i-1})} + 
\lambda_3 \cdot \frac{c(y_i)}{M}
\]

\[
\sum_i \lambda_i = 1, \lambda_i \geq 0
\]

\[
e(x \mid y) = \frac{c(x, y)}{c(y)}
\]

Transition

Emission
Viterbi Decoding (Reminder)

**Input:** a sentence $x_1, \ldots, x_n$, parameters $q, e$, and tag set $S$

**Base case:** $\pi(0, *, *) = 1$

**Definition:** $S_{-1} = S_0 = \{*\}, S_k = S$ for $k \in \{1 \ldots n\}$

**Algorithm:**

for $k \in \{1 \ldots n\}, u \in S_{k-1}, v \in S_k$ :

$$\pi(k, u, v) = \max_{w \in S_{k-2}} \pi(k - 1, w, u) \times q(v | w, u) \times e(x_k | v)$$

$$bp(k, u, v) = \arg \max_{w \in S_{k-2}} \pi(k - 1, w, u) \times q(v | w, u) \times e(x_k | v)$$

$$(y_{n-1}, y_n) = \arg \max_{u,v} (\pi(n, u, v) \times q(\text{STOP} | u, v))$$

for $k = (n - 2) \ldots 1, y_k = bp(k + 2, y_{k+1}, y_{k+2})$

return $y_1, \ldots, y_n$
Log-linear models

- **Input**: domain $X$, Input label set $Y$

- **Feature function**:
  \[
  f : X \times Y \rightarrow \mathbb{R}^m
  \]

- **Parameters**:
  \[
  \theta \in \mathbb{R}^m
  \]

- **Output**: $p(y \mid x)$
  \[
  p_\theta(y \mid x) = \frac{e^{f(x,y)\,^\top\theta}}{\sum_{y' \in Y} e^{f(x,y')\,^\top\theta}}
  \]
Tagging

• We learned to estimate \( p(w, t) \)

• We want to define a model that directly estimates \( p(t | w) \)
  
  • Why?

\[
p(t_1, \ldots, t_n \mid w_1, \ldots, w_n, t_{-1}, t_0) = p(t_1 \mid t_{-1}, t_0, w_1, \ldots, w_n) \\
\times p(t_2 \mid t_{-1}, t_0, t_1, w_1, \ldots, w_n) \\
\vdots \\
\times \ldots p(t_n \mid t_{-1}, \ldots, t_{n-1}, w_1, \ldots, w_n) \\
= \prod_{i=1}^{n} p(t_i \mid t_{-1}, \ldots, t_{i-1}, w_1, \ldots, w_n)
\]
What are X and Y

• We want a model

\[ p(t_i \mid t_1, \ldots, t_{i-1}, w_1, \ldots, w_n) \]

Hispaniola/NNP quickly/RB became/VB an/DT important/JJ base/?? from which Spain expanded its empire into the rest of the Western Hemisphere.

• This is a multi-class problem with classes NN, NNS, Vt, Vi, IN, DT, …
Features

• A feature is a function from \((x, y)\) to a real value:

\[
  f_k(x, y) \in \mathbb{R} \\
  f_k(x, y) \in \{0, 1\}
\]

• A feature vector holds \(m\) features

\[
f(x, y) = \langle f_1(x, y), f_2(x, y), \ldots, f_m(x, y) \rangle
\]
Tagging features

\[ x = \langle t_1, \ldots, t_{i-1}, w_1, \ldots, w_n, i \rangle \]

\[ y \in S \quad (\text{NN, NNS, \ldots}) \]

\[ f(x, y) \in \mathbb{R}^m \]

\[ f_1(x, y) = \begin{cases} 
1 & \text{if } w_i = \text{base and } y = \text{Vt} \\
0 & \text{otherwise} 
\end{cases} \]

\[ f_1(x, y) = \begin{cases} 
1 & \text{if } w_i \text{ ends in “ing” and } y = \text{VBG} \\
0 & \text{otherwise} 
\end{cases} \]

- Use feature templates for pairing all word/tag pairs or all suffix/tag pairs
Feature representation

- Feature vectors are sparse - mostly zeros

- How many features are on in a LM with trigram, bigram and unigram feature templates?

- Represent with hash maps (or hash sets if all features are binary):

\[
f(\langle \ldots, DT, JJ, Hispaniola, \ldots, 6 \rangle, Vt) = \{121, 1003, 21554\}
f(\langle \ldots, DT, JJ, Hispaniola, \ldots, 6 \rangle, JJ) = \{3, 115, 2051\}
\]

- Efficient in memory and dot-product computation (f’ is a dense representation and f” is a sparse one)

\[
f(x, y) \top \theta = \sum_k f'_k(x, y) \cdot \theta_k = \sum_{k \in f''_k(x, y)} \theta_k
\]
Conditional model

- We use the features to define a score for every possible label

\[ s(x, y) = f(x, y) \top \theta = \sum_{k} f_k(x, y) \cdot \theta_k, \quad \theta \in \mathbb{R}^m, f(x, y) \in \mathbb{R}^m \]

- We can now define a conditional probability \( p(y \mid x) \)

\[
p_{\theta}(Y = y \mid x) = \frac{e^{s(x, y)}}{\sum_{y' \in \mathcal{Y}} e^{s(x, y')}} = \text{softmax}(s(x, y_1), s(x, y_2), \ldots s(x, y_{|\mathcal{Y}|}))
\]

\[
(1, 2, 3, 4) \rightarrow \left( \frac{e^1}{e^1 + e^2 + e^3 + e^4}, \frac{e^2}{e^1 + e^2 + e^3 + e^4}, \frac{e^3}{e^1 + e^2 + e^3 + e^4}, \frac{e^4}{e^1 + e^2 + e^3 + e^4} \right)
\]

\[
= (0.032, 0.087, 0.237, 0.644)
\]
Relation to deep learning

• We saw the softmax layer as the final layer in multi-class neural networks.

• Log-linear models:
  • Use human knowledge to write down features that will allow a linear decision boundary in a high-dimensional space.

• Deep learning models:
  • Learn a low-dimensional representation (features) with a neural net that will allow a linear decision boundary
Why log-linear? well...

\[
\log p_\theta(y \mid x) = \log e^{f(x,y)^\top \theta} - \log \sum_{y' \in \mathcal{Y}} e^{f(x,y')^\top \theta} = f(x, y)^\top \theta - \log \sum_{y' \in \mathcal{Y}} e^{f(x,y')^\top \theta}
\]

- First term is linear in parameters
- Second term does not depend on \( y \)
Parameter estimation
Maximum likelihood

• Input: training examples

\[ \{(x^{(i)}, y^{(i)})\}_{i=1}^{n}, \quad (x^{(i)}, y^{(i)}) \in \mathcal{X} \times \mathcal{Y} \]

• Objective function:

\[
L(\theta) = \sum_{i=1}^{n} \log p_\theta(y^{(i)} | x^{(i)}) \\
= \sum_{i=1}^{n} f(x^{(i)}, y^{(i)})^\top \theta - \sum_{i=1}^{n} \log \sum_{y' \in \mathcal{Y}} e^{f(x^{(i)}, y')^\top \theta} \\
\theta_{\text{ML}} = \arg \max_{\theta} L(\theta)
\]
Optimization

- L is a concave function (sum of concave functions)
- Can find optimal point with gradient-based methods
Computing the gradient

- Let’s look at one example

\[
\frac{\partial L^{(i)}(\theta)}{\theta_k} = f_k(x^{(i)}, y^{(i)}) - \frac{1}{\sum_{z' \in Y} e^{f(x^{(i)}, z')^\top \theta}} \cdot \frac{\partial (\sum_{y' \in Y} e^{f(x^{(i)}, y')^\top \theta})}{\partial \theta_k} \\
= f_k(x^{(i)}, y^{(i)}) - \sum_{y' \in Y} f_k(x^{(i)}, y') \cdot \frac{e^{f(x^{(i)}, y')^\top \theta}}{\sum_{z' \in Y} e^{f(x^{(i)}, z')^\top \theta}} \\
= f_k(x^{(i)}, y^{(i)}) - \sum_{y' \in Y} f_k(x^{(i)}, y') \cdot p_{\theta}(y' | x) \\
= f_k(x^{(i)}, y^{(i)}) - \mathbb{E}_{y' \sim p_{\theta}} [f_k(x, y')] 
\]

- Add empirical count and subtract expected counts under the model
SGD vs. Perceptron

- Input: \((x, y)\)

- **SGD**
  \[
  \theta_{\text{new}} = \theta_{\text{old}} + \eta \cdot (f(x, y) - \mathbb{E}_{y' \sim p_{\theta_{\text{old}}}}[f(x, y')])
  \]

- **Perceptron**
  \[
  \theta_{\text{new}} = \theta_{\text{old}} + f(x, y) - f(x, \arg \max_{y'} f(x, y')^\top \theta_{\text{old}})
  \]

- What are the differences? In practice Averaged perceptron is usually used.
Regularization

- Say the word “base” appears exactly twice in the training data and it is tagged as Vt in both times

\[ f_{100}(x, y) = \begin{cases} 
1 & \text{if } w_i = \text{“base” and } y = \text{Vt} \\
0 & \text{otherwise} 
\end{cases} \]

- The gradient with respect to \( \theta_{100} \) are zero when:

\[
\sum_{i=1}^{n} f_{100}(x^{(i)}, y^{(i)}) = \sum_{i=1}^{n} \sum_{y' \in Y} p_{\theta}(y' \mid x) \cdot f_{100}(x^{(i)}, y')
\]

- This happens when \( \theta_{100} \to \infty \): will not affect any other training example ==> bad generalization

- We will always tag base as Vt for any history
L2 regularization

\[ L(\theta) = \sum_{i=1}^{n} \log p_{\theta}(y^{(i)} | x^{(i)}) + \lambda \cdot ||\theta||^2 \]

- Penalize large parameter values
- Empirically this allows using millions of features
Example for language modeling

- With n-gram features with no regularization:
  - The solution is exactly the ML estimate we saw in generative models

- With regularization
  - Empirical performance is at least as good as discounting methods
Independence assumption

• Markov assumption again:

\[
p(t_1, \ldots, t_n \mid w_1, \ldots, w_n, t_{-1}, t_0) = \prod_{i=1}^{n} p(t_i \mid t_{-1}, \ldots, t_{i-1}, w_1, \ldots, w_n)
\]

\[
= \prod_{i=1}^{n} p(t_i \mid t_{i-2}, t_{i-1}, w_1, \ldots, w_n)
\]

• Question: what will we lose if we don’t make this assumption?

• This is called a **locally normalized** log-linear model

• Why?
Example

Hispaniola/NNP quickly/RB became/VB an/DT important/JJ base/?? from which Spain expanded its empire into the rest of the Western Hemisphere.

• A **history** $x$ is a 4-tuple <$t_{i-2}$, $t_{i-1}$, $w$, $i$>

• A **label** $y$ is $t_i$.

  $t_{i-2} : DT$
  $t_{i-1} : JJ$
  $w : Hispaniola ... Hemisphere$
  $i : 6$
  $t_i : NN$
Training

• Given training set of sentences and tag sequences \( w(j), t(j), j = 1...n \)

• Build a training set of the form \( x(i), y(i) \) by constructing all history/label pairs.

• Maximize the L2-regularized maximum-likelihood estimate

• Gradient is the difference between the empirical expected feature vector and model expected feature vector

\[
L(\theta) = \sum_{i=1}^{n} \log p_{\theta}(y^{(i)} | x^{(i)}) + \lambda \cdot ||\theta||^2
\]
Decoding
Greedy

• Greedy decoding
  • For $i = 1 \ldots n$
    • Choose $t_i$ with maximal probability
  • This failed in generative models, but works pretty well here. Why?
  • Complexity?
  • Can we drop some independence assumptions?
Beam search

- Beam search
  - Keep a set of $K$ current hypotheses
  - For $i = 1 \ldots n$
    - Consider all continuations $t_i$ of current set of hypotheses $t_1 \ldots t_{i-1}$ and score them
  - Keep the top $K$

Fed raises interest rates 0.5 percent
Viterbi

• After training the log-linear model we obtain:

\[ p_\theta(y | x) = p_\theta(t_i | t_{i-2}, t_{i-1}, w, i) \]

• Our goal in decoding is to solve:

\[ \arg \max_{t_1, \ldots, t_n} \prod_{i=1}^{n} p(t_i | t_{i-2}, t_{i-1}, w, i) \]

• Solution: Features depend on local hidden variables and so we can use Viterbi
Viterbi

- Define the probability of a tag prefix

\[ r(t_1, \ldots, t_k) = \prod_{i=1}^{k} p_\theta(t_i \mid t_{i-2}, t_{i-1}, w, i) \]

- Define the dynamic programming chart/table:

- Maximum probability for a tag sequence of length \( k \) ending in \( u, v \)

\[ \pi(k, u, v) = \max_{t_1, \ldots, t_{k-2}} r(t_1, \ldots, t_k) \]
Viterbi

**Definition:** $S_k$ is the set of possible tags in position

**Base:** $\pi(0, *, *) = 1$ for all $k \in \{1 \ldots n\}$, for all $u \in S_{k-1}, v \in S_k$:

$$\pi(k, u, v) = \max_{t \in S_{k-2}} \pi(k-1, t, u) \times q(v | t, u, w, k)$$

- **Correctness:** again any sequence ending in triples $t, u, w$ in position $k$, must go through the highest probability sequence ending in $t, u$ in position $k-1$
- **Space and time complexity:** just like HMMs
  - Time $O(n|S|^3)$, Space $O(n|S|^2)$
Add backpointers

**Definition:** $S_k$ is the set of possible tags in position $k$.

**Base:** $\pi(0, *, *) = 1$

for all $k \in \{1 \ldots n\}$, for all $u \in S_{k-1}, v \in S_k$:

$$\pi(k, u, v) = \max_{t \in S_{k-2}} \pi(k-1, t, u) \times q(v | t, u, w, k)$$

$$bp(k, u, v) = \arg \max_{t \in S_{k-2}} \pi(k-1, t, u) \times q(v | t, u, w, k)$$

$$(y_{n-1}, y_n) = \arg \max_{u, v} \pi(n, u, v)$$

for $k = (n - 2) \ldots 1$, $y_k = bp(k + 2, y_{k+1}, y_{k+2})$
Ratnaparkhi, 1996

• Feature templates for
  
  • word/tag pairs
  
  • prefixes <5/tag pairs
  
  • suffixes <5/tag pairs
  
  • tag trigrams
  
  • tag bigrams
  
  • tag unigrams
  
  • word/tag pairs for previous word
  
  • word/tag pairs for subsequent word
Greedy vs. Viterbi
Greedy vs. Viterbi

• What are the advantages of using greedy?
  • Faster
  • Feature function not constrained to the previous two tags

• What are the advantages of Viterbi
  • Global inference
  • Does not suffer from error propagation at test time
### Greedy vs. Viterbi

<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 Chinese/Treebank 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 Beck 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 Sintáctica/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>

\[294^2 = 86436\]
\[294^4 = 7471182096\]
\[45^2 = 2045\]
\[45^4 = 4100625\]

\[11^2 = 121\]
\[11^4 = 14641\]

[Petrov et al. 2012]
<table>
<thead>
<tr>
<th></th>
<th>Token-level accuracy</th>
<th>OOV accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>HMM</strong></td>
<td>96.46%</td>
<td>85.86%</td>
</tr>
<tr>
<td><strong>MEMM</strong></td>
<td>96.96%</td>
<td>91.29%</td>
</tr>
</tbody>
</table>
MEMM vs. HMM

• McCallum et al. (2000) compared the two in a task of FAQ segmentation
  • Modeling word generation is hard
  • Tags: head, question, answer

2.6) What configuration of serial cable should I use

Here follows a diagram of the necessary connections
programs to work properly. They are as far as I know t
agreed upon by commercial comms software developers fo
Pin 1, 4, and 8 must be connected together inside
is to avoid the well known serial port chip bugs. The
MEMM

• Current tag is $t$, previous tag is $t'$ and
  • Line begins with number/punctuation/wh-word…
  • Line contains …
  • Line ends with …
  • Line indentation is
Token HMM

- Generate each word in the sentence independently from other words given the tag seems bad

\[
p(\text{"2.6) What configuration..."}| \text{question}) = p(\text{"2.6"} | \text{question}) \times p(\text{"What"} | \text{question}) \times p(\text{"configuration"} | \text{question}) \times \ldots
\]
Feature HMM

• Replace words with features and then generate:

\[
p\left(\text{“2.6) What configuration...”} \mid \text{question}\right) = p(\text{begins-with-number} \mid \text{question}) \times p(\text{wh-word} \mid \text{question}) \times p(\text{contains-alphanum} \mid \text{question}) \times \ldots
\]
# FQA results

<table>
<thead>
<tr>
<th>Method</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>ME-stateless</td>
<td>0.038</td>
<td>0.362</td>
</tr>
<tr>
<td>Token HMM</td>
<td>0.276</td>
<td>0.140</td>
</tr>
<tr>
<td>Feature HMM</td>
<td>0.413</td>
<td>0.529</td>
</tr>
<tr>
<td>MEMM</td>
<td>0.867</td>
<td>0.681</td>
</tr>
</tbody>
</table>

- **Precision**: number of correct segments predicted divided by total number of segments predicted
- **Recall**: Number of correct segments predicted divided by total number of true segments

Rich overlapping features help a lot!
Model zoo

<table>
<thead>
<tr>
<th></th>
<th>Tagging</th>
<th>Parsing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generative</td>
<td>HMMs</td>
<td>PCFGs</td>
</tr>
<tr>
<td>Log-linear</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Greedy</td>
<td></td>
<td>Transition-based parsing</td>
</tr>
<tr>
<td>Locally normalized</td>
<td>Viterbi</td>
<td>CKY</td>
</tr>
<tr>
<td>Globally normalized</td>
<td>forward-backward</td>
<td>Inside-outside</td>
</tr>
</tbody>
</table>

And then also deep learning variants!
Summary

• Decompose a tag sequence to a sequence of "decisions"

• Train a log-linear model for making local decisions

• Decode with Viterbi
  • Or greedy

• Or beam search