Sequence Models 2

CS 287

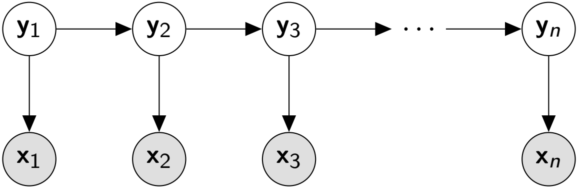
Review: NER Tagging

- **B-TYPE** Stop current mention and begin new mention
- **I-TYPE** Continue adding to current mention
- **O** Not part of a mention.

Example:

`[PER George Bush ] [LOC U.S. ] president is traveling to [LOC Baghdad ] .`

Review: Hidden Markov Model



Review: Hidden Markov Model

Hidden Markov model requires two distributions,

- Transition distribution

$$p(\mathbf{y}_i | \mathbf{y}_{i-1}; \theta)$$

- Emission distribution

$$p(\mathbf{x}_i | \mathbf{y}_i; \theta)$$

- How many total parameters?

Structured Classification with HMM

- Interested in find the most-likely $\mathbf{y}$ conditioned on input $\mathbf{x}$
- As with naive Bayes, can use joint instead.

$$
\begin{aligned}
\arg\max_{c_{1:n}} p(\mathbf{y}_{1:n} = c_{1:n} | \mathbf{x}_{1:n}) &= \arg\max_{c_{1:n}} \log p(\mathbf{y}_{1:n} = c_{1:n}, \mathbf{x}_{1:n}) \\
&= \arg\max_{c_{1:n}} \sum_{i=1}^{n} \log p(\mathbf{y}_i | \mathbf{y}_{i-1}) + \log p(\mathbf{x}_i | \mathbf{y}_i)
\end{aligned}
$$

Review: Maximum Entropy Markov Model

MEMM estimates only a transition distribution,

- Transition distribution (also conditioned on input)

$$p(\mathbf{y}_i | \mathbf{y}_{i-1} = \delta(c_{i-1}), \mathbf{x}_1, \ldots, \mathbf{x}_n) = \mathrm{softmax}(\mathit{feat}(\mathbf{x}, c_{i-1})\mathbf{W} + \mathbf{b})$$

- *feat*; combination of the input and the previous c_{i-1}

Structured Classification with MEMM

- Interested in find the most-likely $\mathbf{y}$ conditioned on input $\mathbf{x}$

$$
\begin{aligned}
\arg\max_{c_{1:n}} p(\mathbf{y}_{1:n} = c_{1:n} | \mathbf{x}_{1:n}) &= \arg\max_{c_{1:n}} \log p(\mathbf{y}_{1:n} = c_{1:n} | \mathbf{x}_{1:n}) \\
&= \arg\max_{c_{1:n}} \sum_{i=1}^{n} \log p(\mathbf{y}_i | \mathbf{y}_{i-1}, \mathbf{x})
\end{aligned}
$$

Markov Models

- In general, intractable to solve sequence prediction,

$$\arg\max_{c_{1:n}} f(\mathbf{x}, c_{1:n})$$

- Today, focus on (first-order) Markov models,

$$f(\mathbf{x}, c_{1:n}) = \sum_{i=1}^{n} \log \hat{\mathbf{y}}(c_{i-1})_{c_i}$$

- Can extend these ideas to higher-order models.

$$f(\mathbf{x}, c_{1:n}) = \sum_{i=1}^{n} \log \hat{\mathbf{y}}(c_{i-2}, c_{i-1})_{c_i}$$

Quiz: History-Based Models

Given this definition of a history-based model,

$$f(\mathbf{x}, c_{1:n}) = \sum_{i=1}^{n} \log \hat{\mathbf{y}}(c_{i-1})_{c_i}$$

Describe the function g for the following models,

1. Hidden Markov Model
2. Maximum-Entropy Markov Model
3. Bigram Language Model (with no $\mathbf{x}$, e.g. best n babble)
4. NNLM with $d_{\mathrm{win}} = 1$

Answers I

- HMM

$$\begin{aligned}\log \hat{\mathbf{y}}(c_{i-1})_{c_i} &= \log p(\mathbf{y}_i = \delta(c_i) | \mathbf{y}_{i-1} = \delta(c_{i-1})) + \log p(\mathbf{x}_i | \mathbf{y}_i) \\ &= \log T_{c_{i-1}, c_i} + \log E_{x_i, c_i}\end{aligned}$$

- MEMM

$$\log \hat{\mathbf{y}}(c_{i-1}) = \log \mathrm{softmax}(\mathit{feat}(\mathbf{x}, c_{i-1})\mathbf{W} + \mathbf{b})$$

- Bigram

$$\log \hat{\mathbf{y}}(c_{i-1})_{c_i} = \log p(\mathbf{y}_i = \delta(c_i) | \mathbf{y}_{i-1} = \delta(c_{i-1}))$$

- NNLM

$$\log \hat{\mathbf{y}}(c_{i-1}) = \log \mathrm{softmax}(\tanh(v(c_{i-1})\mathbf{W}^1 + \mathbf{b}^1)\mathbf{W}^2 + \mathbf{b}^2)$$

Today's Lecture

Search for Sequences (Fun with Factoring?)

$$f(\mathbf{x}, c_{1:n}) = \sum_{i=1}^{n} \log \hat{\mathbf{y}}(c_{i-1})_{c_i}$$

- Greedy Search
- Beam Search
- Viterbi Search

Structured Perceptron

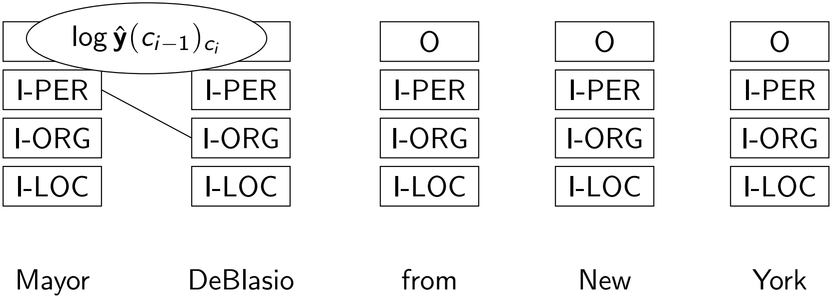
The Lattice

$$f(\mathbf{x}, c_{1:n}) = \sum_{i=1}^{n} \log \hat{\mathbf{y}}(c_{i-1})_{c_i}$$

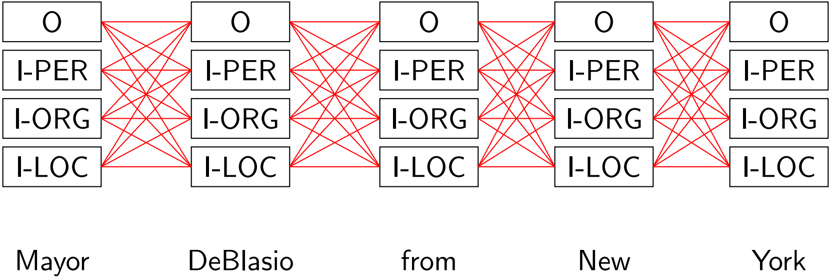
O	O	O	O	O
I-PER	I-PER	I-PER	I-PER	I-PER
I-ORG	I-ORG	I-ORG	I-ORG	I-ORG
I-LOC	I-LOC	I-LOC	I-LOC	I-LOC
Mayor	DeBlasio	from	New	York

The Lattice

$$f(\mathbf{x}, c_{1:n}) = \sum_{i=1}^{n} \log \hat{\mathbf{y}}(c_{i-1})_{c_i}$$

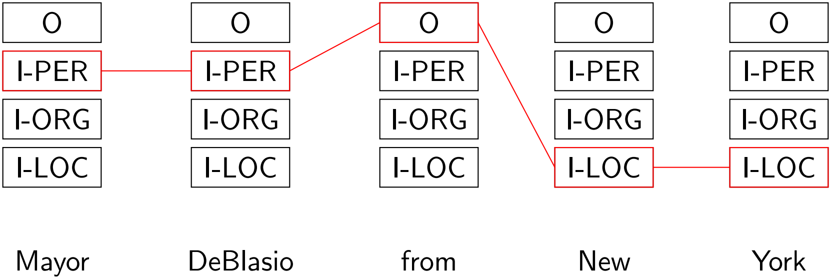


Exponentially Many Sequences



A Single Sequence

$$f(\mathbf{x}, c_{1:n}) = \log \hat{\mathbf{y}}(c_1)_{c_2} + \log \hat{\mathbf{y}}(c_2)_{c_3} + \log \hat{\mathbf{y}}(c_3)_{c_4} + \log \hat{\mathbf{y}}(c_4)_{c_5}$$



Contents

Heuristic Search

- Fast method for finding a solution.
- Often can be quite effective in practice.
- Tradeoff: More powerful models/less exact search.

Algorithm 1: Greedy Search

procedure GREEDYSEARCH

$s = 0$ ▷ Running score

$c \in \mathcal{C}^{n+1}$ ▷ Sequence

$c_0 = \langle s \rangle$ ▷ Initial Symbol

for $i = 1$ to n **do**

$$c_i \leftarrow \arg\max_{c'_i} \hat{\mathbf{y}}(c_{i-1})_{c'_i}$$

$$s \leftarrow s + \log \hat{\mathbf{y}}(c_{i-1})_{c_i}$$

return c, s

- Time Complexity?
- Space Complexity?

Algorithm 1: Greedy Search

procedure GREEDYSEARCH

$s = 0$ ▷ Running score

$c \in \mathcal{C}^{n+1}$ ▷ Sequence

$c_0 = \langle s \rangle$ ▷ Initial Symbol

for $i = 1$ to n **do**

$c_i \leftarrow \arg\max_{c'_i} \hat{\mathbf{y}}(c_{i-1})_{c'_i}$

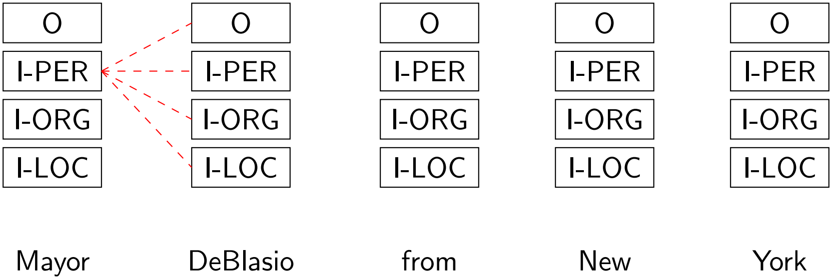
$s \leftarrow s + \log \hat{\mathbf{y}}(c_{i-1})_{c_i}$

return c, s

- Time Complexity?
- Space Complexity?

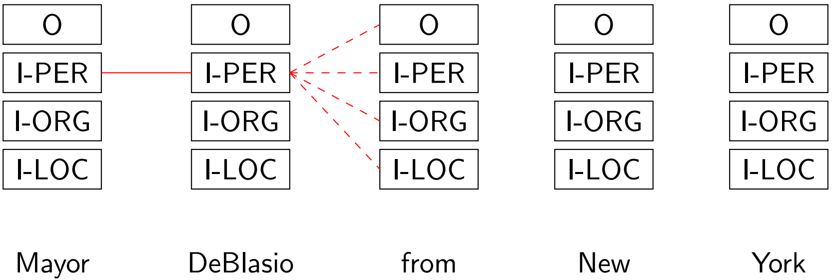
Greedy Search

$$f(\mathbf{x}, c_{1:n}) = \sum_{i=1}^{n} \log \hat{\mathbf{y}}(c_{i-1})_{c_i}$$



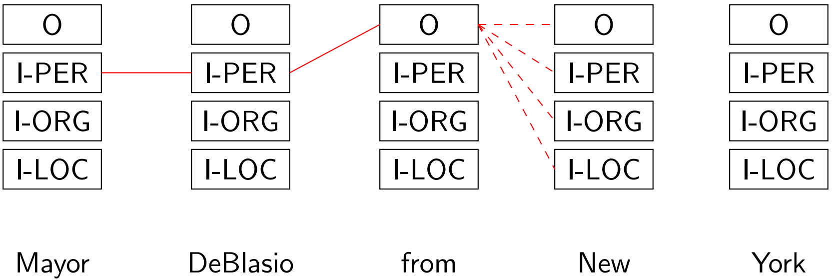
Greedy Search

$$f(\mathbf{x}, c_{1:n}) = \sum_{i=1}^{n} \log \hat{\mathbf{y}}(c_{i-1})_{c_i}$$



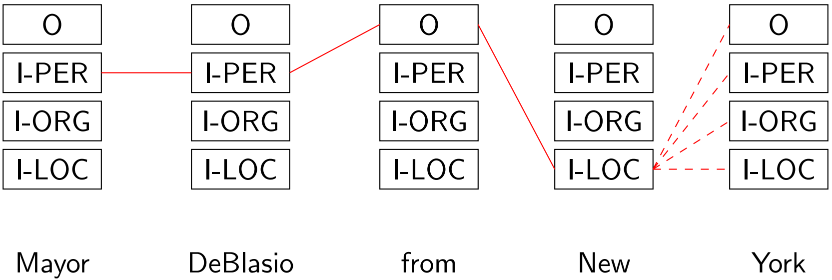
Greedy Search

$$f(\mathbf{x}, c_{1:n}) = \sum_{i=1}^{n} \log \hat{\mathbf{y}}(c_{i-1})_{c_i}$$



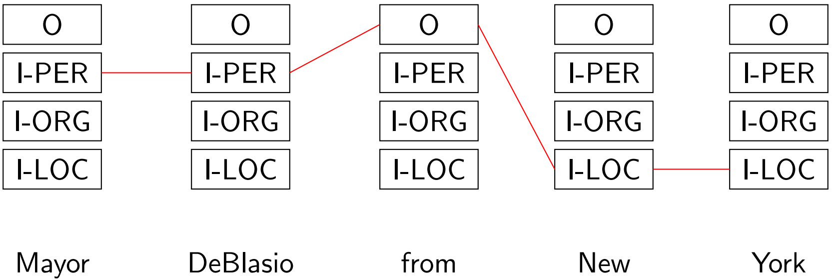
Greedy Search

$$f(\mathbf{x}, c_{1:n}) = \sum_{i=1}^{n} \log \hat{\mathbf{y}}(c_{i-1})_{c_i}$$



Greedy Search

$$f(\mathbf{x}, c_{1:n}) = \sum_{i=1}^{n} \log \hat{\mathbf{y}}(c_{i-1})_{c_i}$$



Issues

- What cases does beam search fail?
- How might we get around these issues without hurting time/space complexity?

Algorithm 2: Beam Search

- Idea: Maintain multiple hypotheses.
- Each step keep k highest scoring.
- Hypotheses with different histories compete.

Algorithm 2: Beam Search

procedure $\textsc{BeamSearch}(K)$

  K sequences $c[1], \dots, c[K]$

  $s_k \leftarrow 0, c[k]_0 = \langle s \rangle$ for all k ▷ Initialize

  for $i = 1$ to n **do**

    hyps $\leftarrow \{\}$

    for $k = 1$ to K **do**

      for $c_i \in \mathcal{C}$ **do**

        $s' \leftarrow s_k + \log \hat{\mathbf{y}}(c[k]_{i-1})_{c_i}$

        hyps add $(c[k] + [c_i], s')$

      for $k = 1$ to K **do**

        $c[k], s_k \leftarrow$ pop highest score in hyps

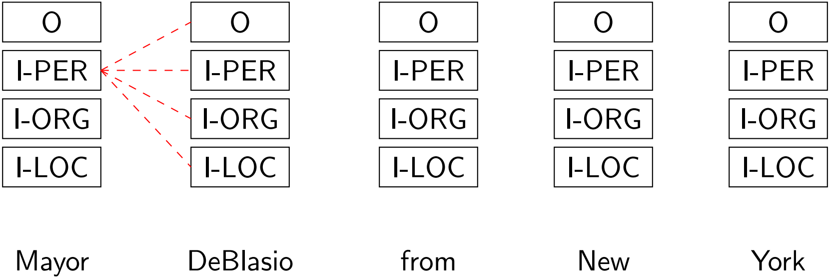
  return $c[1], s_1$

Questions: Beam Search

- Space Complexity?
- Time Complexity?
- Optimality?
- Versus K runs of beam search?

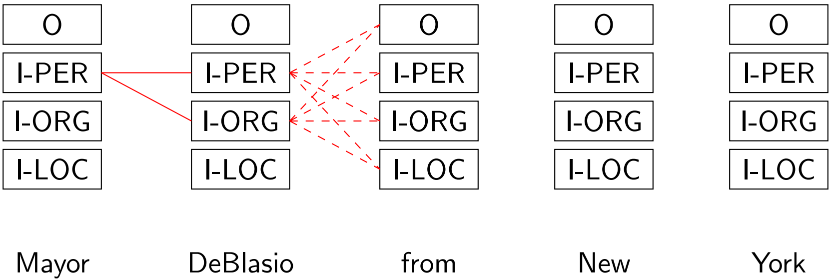
Beam Search (K=2)

$$f(\mathbf{x}, c_{1:n}) = \sum_{i=1}^{n} \log \hat{\mathbf{y}}(c_{i-1})_{c_i}$$



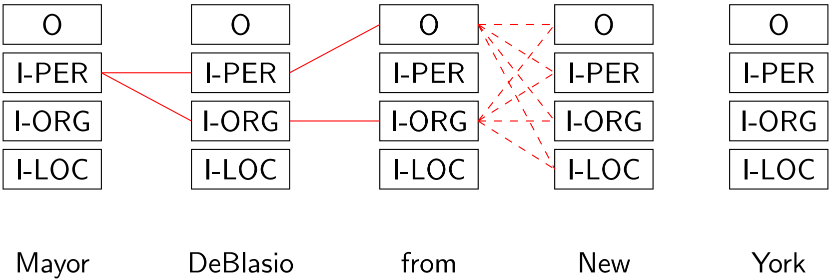
Beam Search (K=2)

$$f(\mathbf{x}, c_{1:n}) = \sum_{i=1}^{n} \log \hat{\mathbf{y}}(c_{i-1})_{c_i}$$



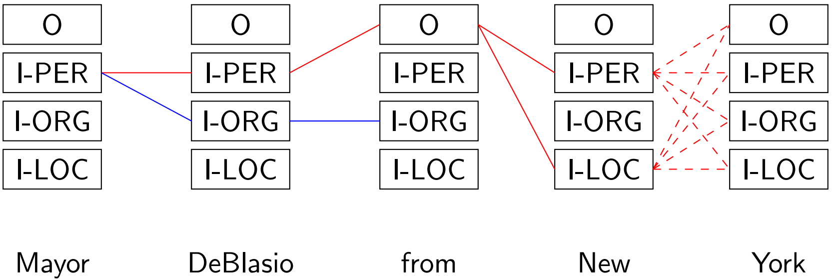
Beam Search (K=2)

$$f(\mathbf{x}, c_{1:n}) = \sum_{i=1}^{n} \log \hat{\mathbf{y}}(c_{i-1})_{c_i}$$



Beam Search (K=2)

$$f(\mathbf{x}, c_{1:n}) = \sum_{i=1}^{n} \log \hat{\mathbf{y}}(c_{i-1})_{c_i}$$

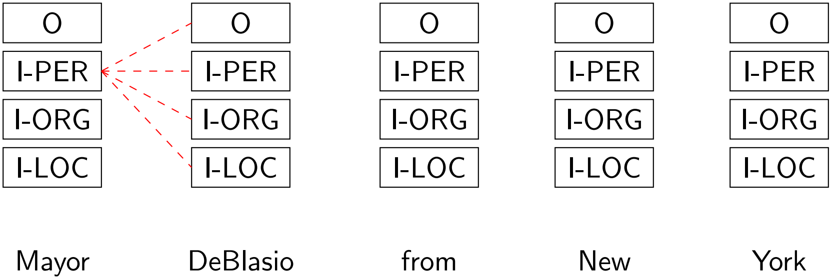


Beam Search for Markov Models

- Does this use the Markov property?
- (How would beam search differ for RNN and HMM?)

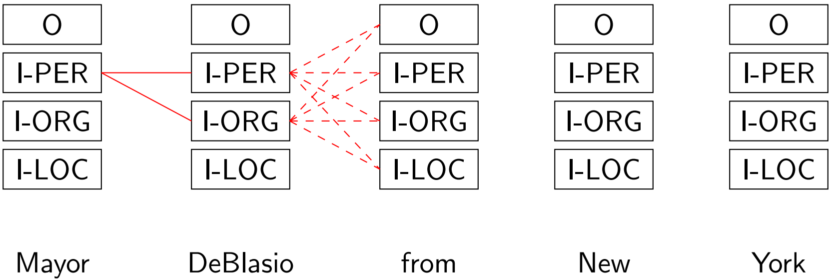
Beam Search

$$f(\mathbf{x}, c_{1:n}) = \sum_{i=1}^{n} \log \hat{\mathbf{y}}(c_{i-1})_{c_i}$$



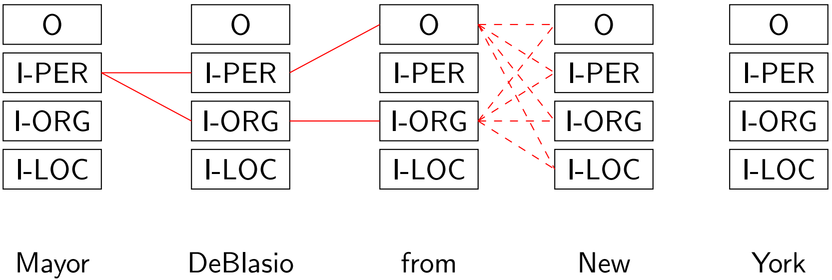
Beam Search

$$f(\mathbf{x}, c_{1:n}) = \sum_{i=1}^{n} \log \hat{\mathbf{y}}(c_{i-1})_{c_i}$$



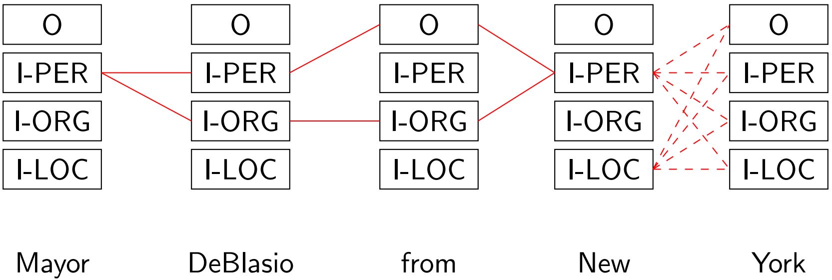
Beam Search

$$f(\mathbf{x}, c_{1:n}) = \sum_{i=1}^{n} \log \hat{\mathbf{y}}(c_{i-1})_{c_i}$$



Beam Search

$$f(\mathbf{x}, c_{1:n}) = \sum_{i=1}^{n} \log \hat{\mathbf{y}}(c_{i-1})_{c_i}$$



Algorithm 2: Beam Search With Recombination

procedure $\textsc{BeamSearch}(K)$

$c \in \mathcal{C}^{K \times n+1}$

$s_k \leftarrow 0, c[k]_0 = \langle s \rangle$ for all k

for $i = 1$ to n **do**

hyps $\leftarrow \{\}$

for $k = 1$ to K **do**

for $c_i \in \mathcal{C}$ **do**

$s' \leftarrow s_k + \log \hat{\mathbf{y}}(c[k]_{i-1})_{c_i}$

hyps add $(c[k] + c_i, s')$

for $k = 1$ to K **do**

$c[k], s_k \leftarrow$ pop highest score in hyps with unique c_i

return $c[1], s_1$

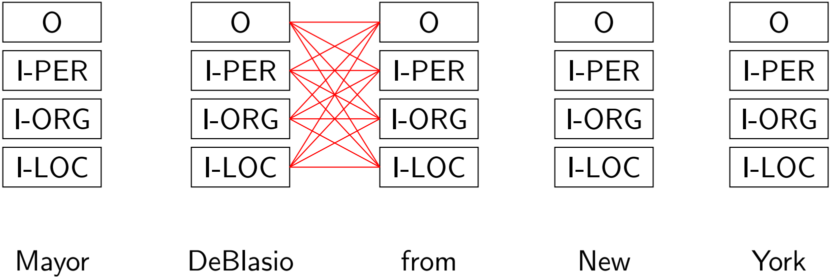
Exact Solutions

- Greedy search and beam search are approximate (heuristic).
- Neither really exploit Markov assumption.
- Exact algorithm for sequences known as Viterbi (1967) decoding

Contents

Dynamic Programming over a Lattice

- Several different varieties: Viterbi, forward, backward
- Recursive Definition:
  - Base Case: Start with the score for sequence of length 1.
  - Inductive Case: Compute all sequences of length i from $i-1$



Viterbi Algorithm (Simple)

procedure $\textsc{Viterbi}$

$\pi \in \mathbb{R}^{(n+1)\times\mathcal{C}}$ initialized to $-\infty$

$\pi[0, \langle s \rangle] = 0$

for $i = 1$ to n **do**

for $c_i \in \mathcal{C}$ **do**

$\pi[i, c_i] = \max_{c_{i-1}} \pi[i-1, c_{i-1}] + \log \hat{\mathbf{y}}(c_{i-1})_{c_i}$

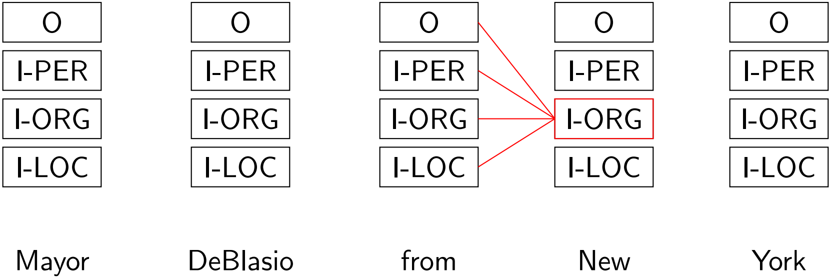
return $\max_{c_n \in \mathcal{C}} \pi[n, c_n]$

- Time Complexity?
- Space Complexity?

The Main Max-Step

for $c_i \in \mathcal{C}$ **do**

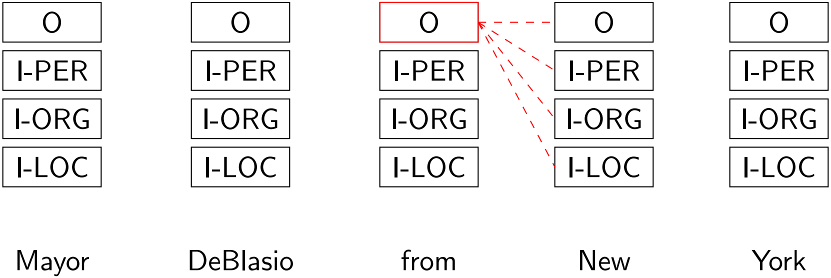
$$\pi[i, c_i] = \max_{c_{i-1}} \pi[i-1, c_{i-1}] + \log \hat{\mathbf{y}}(c_{i-1})_{c_i}$$



Computation and Ordering

for $c_i \in \mathcal{C}$ **do**

$$\pi[i, c_i] = \max_{c_{i-1}} \pi[i-1, c_{i-1}] + \log \hat{\mathbf{y}}(c_{i-1})_{c_i}$$



Viterbi Algorithm with Precompute

procedure $\textsc{ViterbiWithPrecompute}$

$\pi \in \mathbb{R}^{n \times \mathcal{C}}$ initialized to $-\infty$

$\pi[0, \langle s \rangle] = 0$

for $i = 1$ to n **do**

for $c_{i-1} \in \mathcal{C}$ **do**

precompute $\hat{\mathbf{y}}(c_{i-1})$

for $c_i \in \mathcal{C}$ **do**

$score = \pi[i-1, c_{i-1}] + \log \hat{\mathbf{y}}(c_{i-1})_{c_i}$

if $score > \pi[i, c_i]$ **then**

$\pi[i, c_i] = score$

return $\max_{c_n \in \mathcal{C}} \pi[n, c_n]$

Viterbi Algorithm with Backpointers

procedure $\textsc{ViterbiWithBP}$

$\pi \in \mathbb{R}^{n+1 \times \mathcal{C}}$ initialized to $-\infty$

$bp \in \mathcal{C}^{n \times \mathcal{C}}$ initialized to ϵ

$\pi[0, \langle s \rangle] = 0$

for $i = 1$ to n **do**

  for $c_{i-1} \in \mathcal{C}$ **do**

    compute $\hat{\mathbf{y}}(c_{i-1})$

    for $c_i \in \mathcal{C}$ **do**

      $score = \pi[i-1, c_{i-1}] + \log \hat{\mathbf{y}}(c_{i-1})_{c_i}$

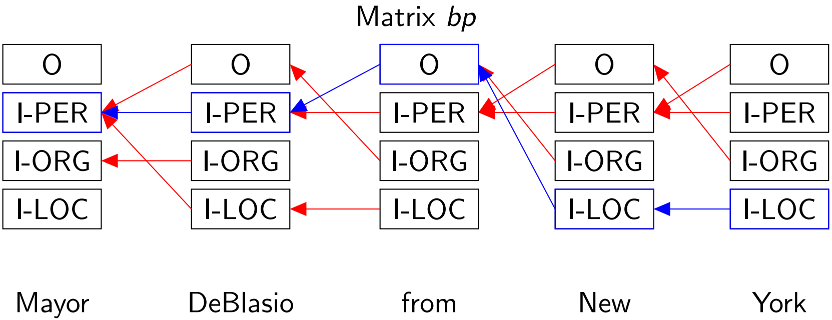
      if $score > \pi[i, c_i]$ **then**

        $\pi[i, c_i] = score$

        $bp[i, c_i] = c_{i-1}$

return sequence from bp

Walking Back



Decoding with MEMM

- Allows features on previous sequence information.
- If $|\mathcal{C}|$ is reasonable, quite fast to decode.
- If $|\mathcal{C}|$ is large, can use approximations (others coming up as well).
- However, still not trained for sequence prediction.

Contents

Recall: Issues with Multiclass for Sequences

- Say there are $\mathcal{T}$ tags and sequence length is n
- There are $d_{\mathrm{out}} = O(\mathcal{T}^n)$ sequences!
- Just naively computing the softmax is exponential in length.
- Even if you could compute the softmax, $\mathbf{W} \in \mathbb{R}^{d_{\mathrm{in}} \times d_{\mathrm{out}}}$ would be impossible to train.

Answers?

- If we have Markov structure, finding max is fast,

$$f(\mathbf{x}, c_{1:n}) = \sum_{i=1}^{n} \log \hat{\mathbf{y}}(c_{i-1})_{c_i}$$

- Can have local features, with small **W** (no softmax)

$$\log \hat{\mathbf{y}}(c_{i-1})_{c_i} = feat(\mathbf{x}, c_{i-1})\mathbf{W} + \mathbf{b}$$

- Can use hinge-loss instead of taking the full softmax.
- (Next class, revisit softmax)

Review: Hinge Loss

$$\mathcal{L}(\theta) = \sum_{i=1}^{n} L_{hinge}(\mathbf{y}, \hat{\mathbf{y}})$$

$$L_{hinge}(\mathbf{y}, \hat{\mathbf{y}}) = \max\{0, 1 - (\hat{y}_c - \hat{y}_{c'})\}$$

Where

- Let c be defined as true class $y_{i,c} = 1$

$$c' = \arg\max_{i \in \mathcal{C} \setminus \{c\}} \hat{y}_i$$

Minimizing hinge loss is an upper-bound for 0/1.

$$L_{hinge}(\mathbf{y}, \hat{\mathbf{y}}) \geq L_{0/1}(\mathbf{y}, \hat{\mathbf{y}})$$

Review: Hinge Loss

$$\mathcal{L}(\theta) = \sum_{i=1}^{n} L_{hinge}(\mathbf{y}, \hat{\mathbf{y}})$$

$$L_{hinge}(\mathbf{y}, \hat{\mathbf{y}}) = \max\{0, 1 - (\hat{y}_c - \hat{y}_{c'})\}$$

Where

- Let c be defined as true class $y_{i,c} = 1$

$$c' = \operatorname*{arg\,max}_{i \in \mathcal{C} \setminus \{c\}} \hat{y}_i$$

Minimizing hinge loss is an upper-bound for $0/1$.

$$L_{hinge}(\mathbf{y}, \hat{\mathbf{y}}) \geq L_{0/1}(\mathbf{y}, \hat{\mathbf{y}})$$

Simple Structured Hinge

$$\mathcal{L}(\theta) = \sum_{i=1}^{n} L_{shinge}(\mathbf{y}, \hat{\mathbf{y}})$$

$$L_{shinge}(\mathbf{y}, \hat{\mathbf{y}}) = \max\{0, 1 - (f(\mathbf{x}, c_{1:n}) - f(\mathbf{x}, c'_{1:n}))\}$$

Where

- Let $c_{1:n}$ be defined as true classes $y_{i,c} = 1$

$$c'_{1:n} = \arg\max_{c' \neq c} f(\mathbf{x}, c'_{1:n})$$

- What do the gradients look like?

Symbolic Gradients

- Let $c_{1:n}$ be defined as true sequence
- Let $c'_{1:n}$ be defined as the highest scoring non-true sequence

$$c'_{1:n} = \arg\max_{c' \neq c} f(\mathbf{x}, c'_{1:n})$$

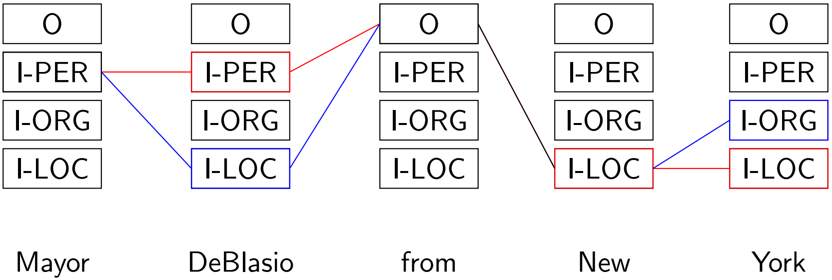
- Partials of $L(y, \hat{y})$

$$\frac{\partial L(y, k\hat{y})}{\partial \log \hat{y}_{i,j}} = \begin{cases} 0 & f(\mathbf{x}, c_{1:n}) - f(\mathbf{x}, c'_{1:n} > 1 \\ 1 & j = c'_i \wedge j \neq c_i \\ -1 & j = c_i \wedge j \neq c'_i \\ 0 & o.w. \end{cases}$$

Intuition: If wrong or close to wrong, improve correct and lower closest incorrect positions (when they disagree).

Updates

- red - c'
- blue - c
- black - both



Structured Perceptron

Similar method without margin

- Let $c_{1:n}$ be defined as true sequence
- Let $c'_{1:n}$ be defined as the highest scoring non-true sequence

$$c'_{1:n} = \arg\max_{c' \neq c} f(\mathbf{x}, c'_{1:n})$$

- Partials of $L(y, \hat{y})$

$$\frac{\partial L(y, k\hat{y})}{\partial \log \hat{y}_{i,j}} = \begin{cases} 0 & f(\mathbf{x}, c_{1:n}) - f(\mathbf{x}, c'_{1:n} > 0 \\ 1 & j = c'_i \wedge j \neq c_i \\ -1 & j = c_i \wedge j \neq c'_i \\ 0 & o.w. \end{cases}$$

- Furthermore, decode with the average of the weights during training (Collins, 2002)

Structured Perceptron versus MEMM (Collins, 2002)