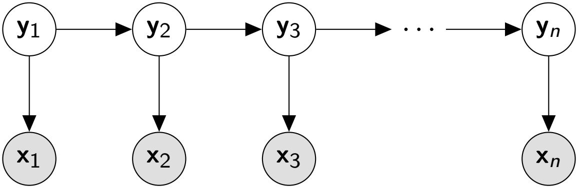
Sequence Models 2

CS 287

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?

Review: Maximum Entropy Markov Model

MEMM estimates only a transition distribution,

- Transition distribution (also conditioned on input)

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

- Here *feat* is a deterministic combination of the input and the previous c_{i-1}
- How many total parameters?

History-Based Model

- In general, intractable to solve sequence prediction,

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

- Today, focus on (first-order) history-based models,

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

- Can extend these ideas to higher-order models.

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

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

- 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

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

- 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

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

Answers

- 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

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

- 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

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

Answers

- 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

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

Answers

- 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

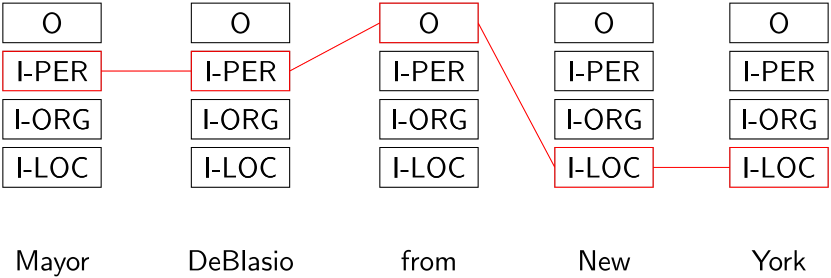
$$\log \hat{\mathbf{y}}(c_{i-1}) = \log \mathrm{softmax}(\mathit{feat}(\mathbf{x}, c_{i-1})\mathbf{W} + \mathbf{b})$$

- Bigram

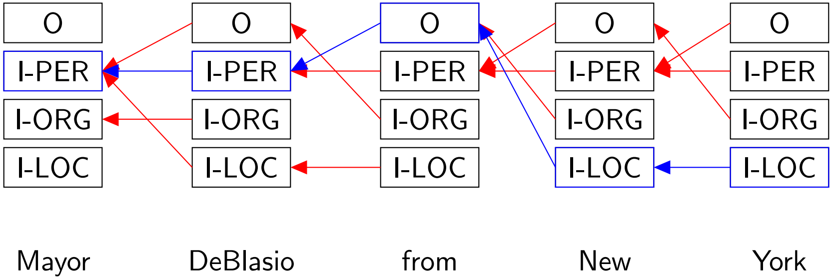
$$\log \hat{\mathbf{y}}(c_{i-1})_{c_i} = \log p(\mathbf{y}_i = \delta(c_i) | \mathbf{y}_{i-1} = \delta(c_{i-1}))$$

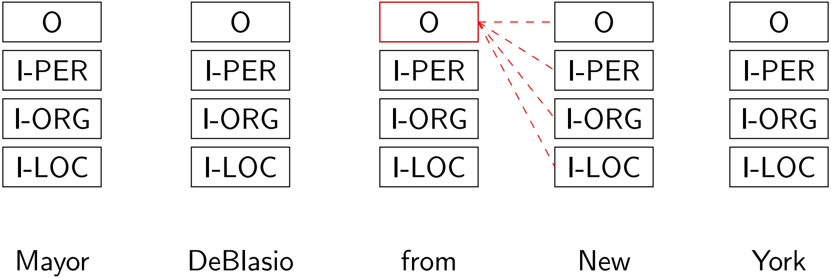
- NNLM

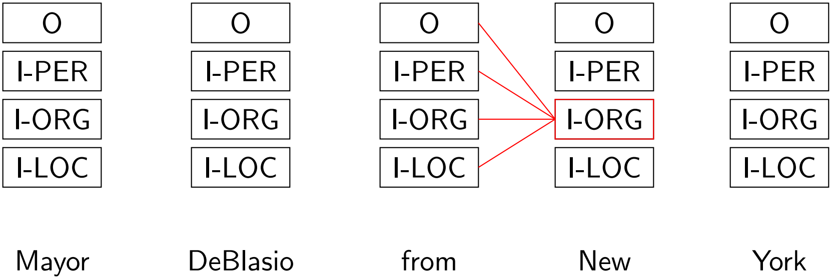
$$\log \hat{\mathbf{y}}(c_{i-1}) = \log \mathrm{softmax}(\tanh(v(c_{i-1})\mathbf{W}^1 + \mathbf{b}^1)\mathbf{W}^2 + \mathbf{b}^2)$$

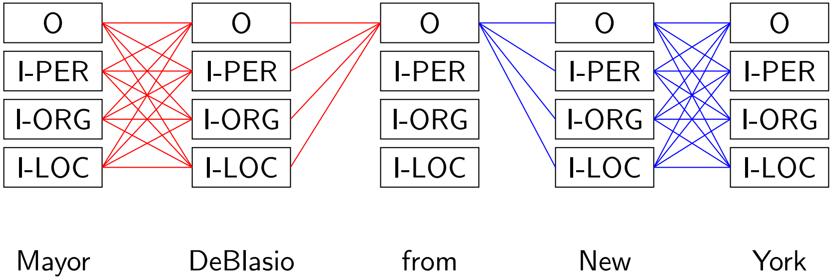


O
I-PER
I-ORG
I-LOC
O
I-PER
I-ORG
I-LOC
O
I-PER
I-ORG
I-LOC
O
I-PER
I-ORG
I-LOC
O
I-PER
I-ORG
I-LOC
Mayor
DeBlasio
from
New
York

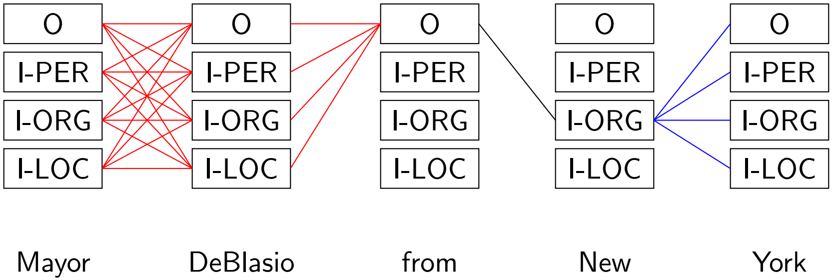

O
I-PER
I-ORG
I-LOC
Mayor
DeBlasio
from
New
York


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


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


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

Edge Marginal



Viterbi Algorithm (Simple)

procedure $\textsc{Viterbi}$

$\pi \in \mathbb{R}^{\{0,\dots,n\} \times \mathcal{C}}$ 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 $\max_{c_n \in \mathcal{C}} \pi[n, c_n]$

Viterbi Algorithm with Precompute

procedure $\textsc{ViterbiWithPrecompute}$

$\pi \in \mathbb{R}^{\{0,\dots,n\}\times\mathcal{C}}$ 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 $\textsc{ViterbiWithBP}$

$\pi \in \mathbb{R}^{\{0,\dots,n\}\times\mathcal{C}}$ initialized to $-\infty$

$bp \in \mathcal{C}^{\{1,\dots,n\}\times\mathcal{C}}$ 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**

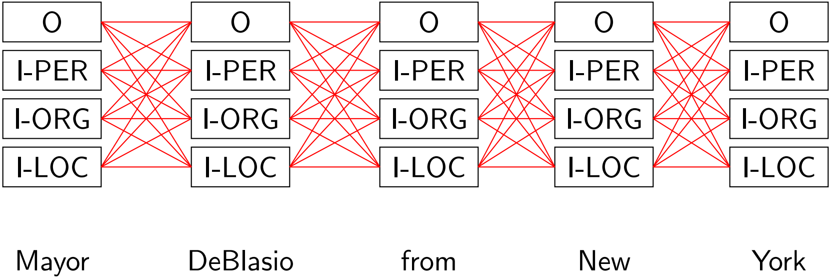
      $score = \pi[i-1, c_{i-1}] + \log \hat{\mathbf{y}}(c_{i-1})_{c_i}$

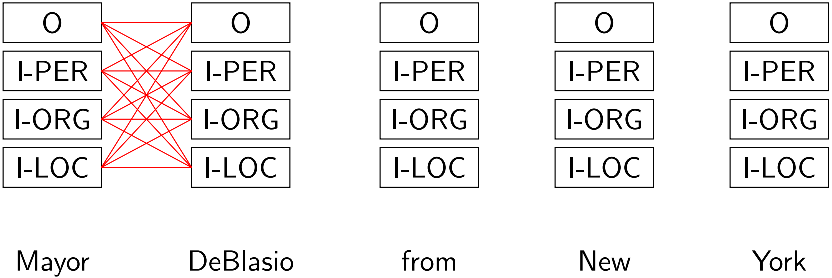
      if $score > \pi[i, c_i]$ **then**

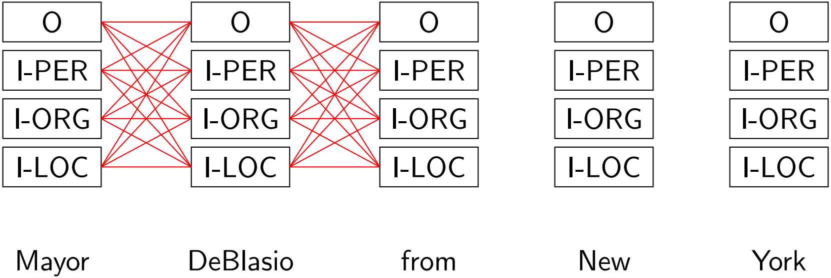
        $\pi[i, c_i] = score$

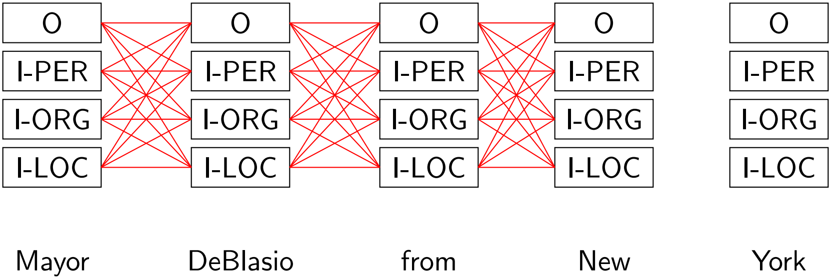
        $bp[i, c_i] = c_{i-1}$

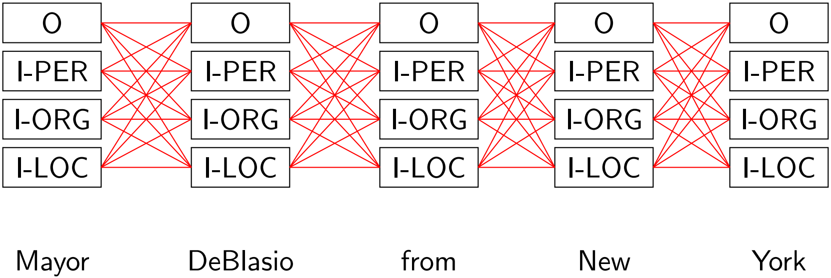
return sequence from bp

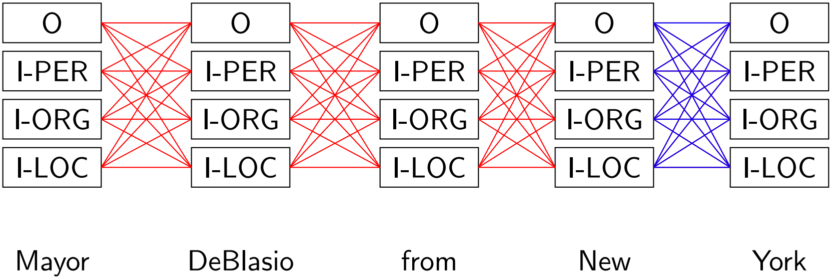

O
I-PER
I-ORG
I-LOC
Mayor
DeBlasio
from
New
York

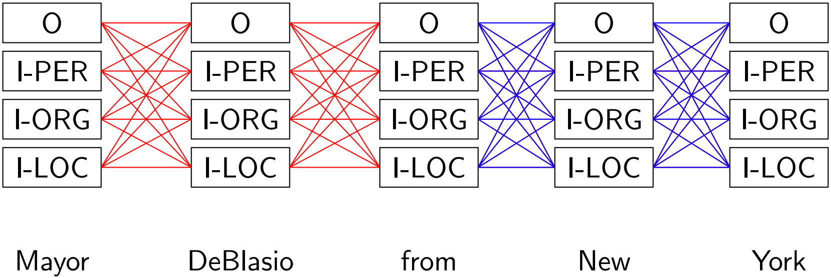

O
I-PER
I-ORG
I-LOC
O
I-PER
I-ORG
I-LOC
O
I-PER
I-ORG
I-LOC
O
I-PER
I-ORG
I-LOC
O
I-PER
I-ORG
I-LOC
Mayor
DeBlasio
from
New
York

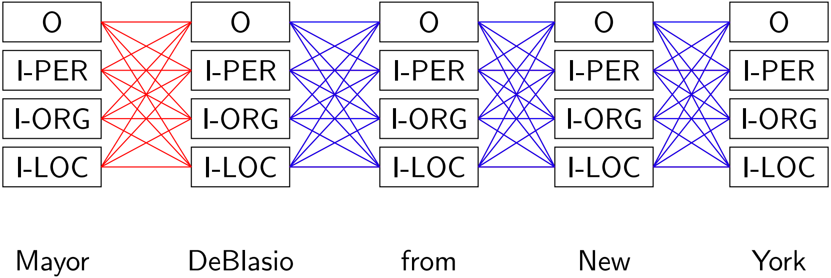

O
I-PER
I-ORG
I-LOC
O
I-PER
I-ORG
I-LOC
O
I-PER
I-ORG
I-LOC
O
I-PER
I-ORG
I-LOC
O
I-PER
I-ORG
I-LOC
Mayor
DeBlasio
from
New
York

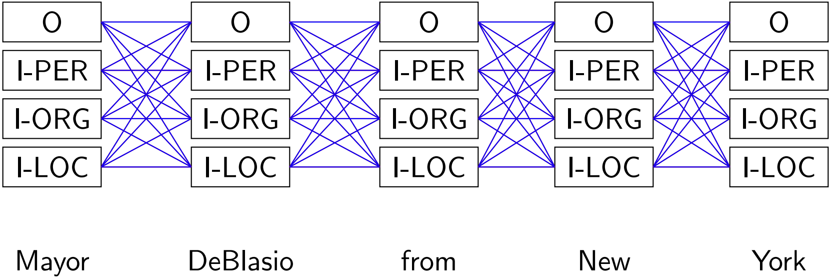

O
I-PER
I-ORG
I-LOC
O
I-PER
I-ORG
I-LOC
O
I-PER
I-ORG
I-LOC
O
I-PER
I-ORG
I-LOC
O
I-PER
I-ORG
I-LOC
Mayor
DeBlasio
from
New
York

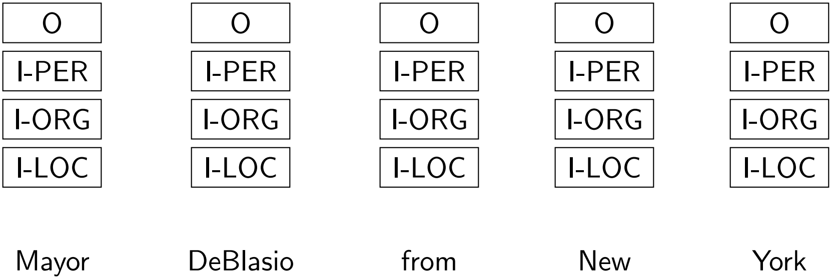

O
I-PER
I-ORG
I-LOC
Mayor
DeBlasio
from
New
York


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

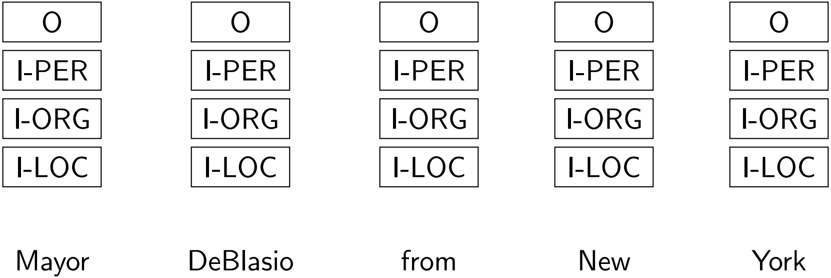

O
I-PER
I-ORG
I-LOC
Mayor
DeBlasio
from
New
York

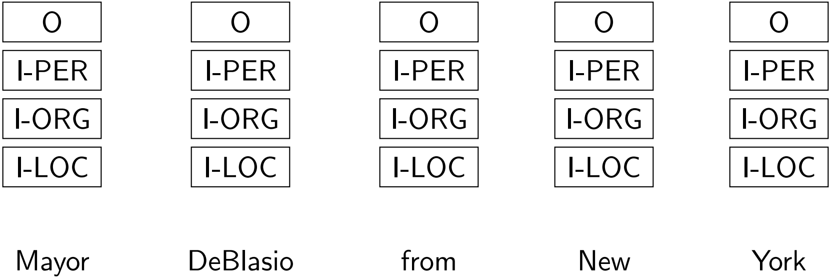

O
I-PER
I-ORG
I-LOC
O
I-PER
I-ORG
I-LOC
O
I-PER
I-ORG
I-LOC
O
I-PER
I-ORG
I-LOC
O
I-PER
I-ORG
I-LOC
Mayor
DeBlasio
from
New
York

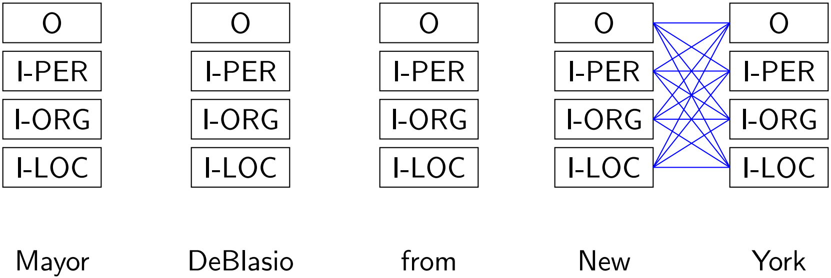

O
I-PER
I-ORG
I-LOC
Mayor
DeBlasio
from
New
York

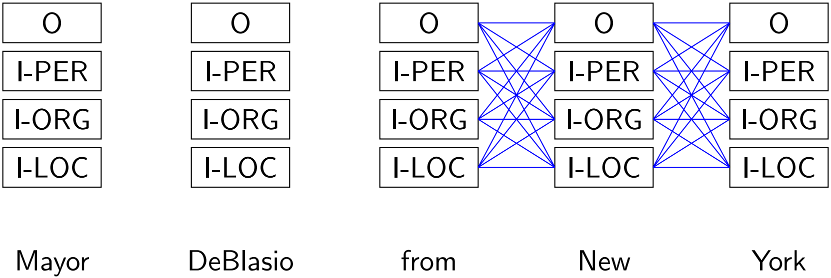

O
I-PER
I-ORG
I-LOC
Mayor
O
I-PER
I-ORG
I-LOC
DeBlasio
O
I-PER
I-ORG
I-LOC
from
O
I-PER
I-ORG
I-LOC
New
O
I-PER
I-ORG
I-LOC
York

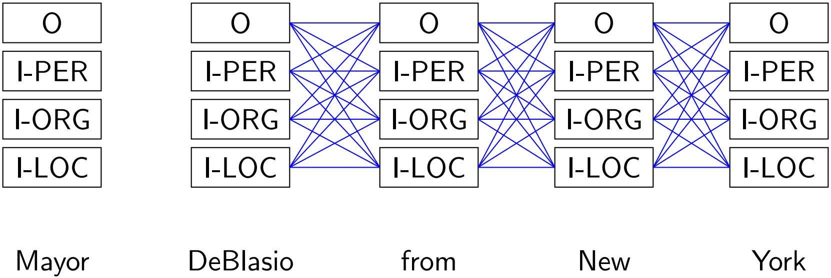
O	O	O	O	O
I-PER	I-PER	I-PER	I-PER	I-PER
I-ORG	I-ORG	I-ORG	I-ORG	I-ORG
I-LOC	I-LOC	I-LOC	I-LOC	I-LOC
Mayor	DeBlasio	from	New	York

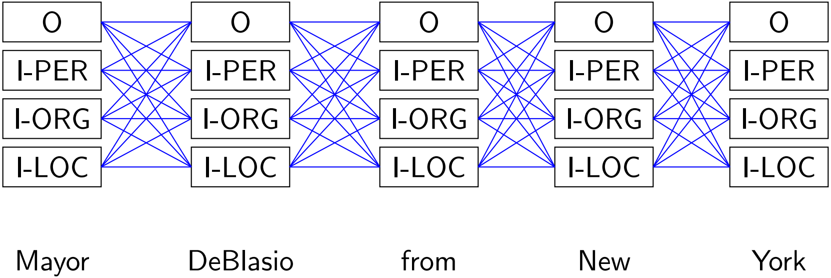

O
I-PER
I-ORG
I-LOC
Mayor
DeBlasio
from
New
York


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


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


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


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


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

Forward Algorithm

procedure FORWARD

$\alpha \in \mathbb{R}^{\{0,\ldots,n\}\times\mathcal{C}}$ initialized to $-\infty$

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

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

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

$\alpha[i, c_i] = \sum_{c_{i-1}} \alpha[i-1, c_{i-1}] * \hat{\mathbf{y}}(c_{i-1})_{c_i}$

return $\sum_{c_n \in \mathcal{C}} \alpha[n, c_n]$

Backward Algorithm

procedure $\textsc{Backward}$

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

$\beta[n+1, \langle s \rangle] = 0$

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

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

    $\beta[i, c_i] = \sum_{c_{i+1}} \beta[i+1, c_{i+1}] * \hat{\mathbf{y}}(c_i)_{c_i+1}$

return $\sum_{c_1 \in \mathcal{C}} \beta[1, c_1]$

Marginals

$$M(c_i) = \sum_{c_{1:n}} f(\mathbf{x}, c_{1:n})$$

$$p(\mathbf{y}_i = c_i | \mathbf{x}) = \sum_{c_i} p(\mathbf{y}_i = \delta(c_i) | \mathbf{x}_{1:n})$$

For the case of MEMM gives you just this.

For HMM

$$\begin{aligned} p(\mathbf{y}_i = c_i | \mathbf{x}) &= \sum_{c_i} p(\mathbf{y}_i = \delta(c_i) | \mathbf{x}_{1:n}) \\ &= p(\mathbf{y}_i = c_i | \mathbf{x}) = \sum_{c_i} p(\mathbf{y}_i = \delta(c_i), \mathbf{x}_{1:n}) / p(\mathbf{x}_{1:n}) (1) \end{aligned}$$

How do you compute $p(\mathbf{x}_{1:n})$?

$$p(\mathbf{x}_{1:n}) = \sum_{c_i}$$

Edge Marginals

$$M(c_{i-1}, c_i) = \sum_{c_{1:n}} f(\mathbf{x}, c_{1:n})$$

Is this the same forward-backward?

Viterbi

Contents