Sequence Models 3

CS 287

Review: 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,

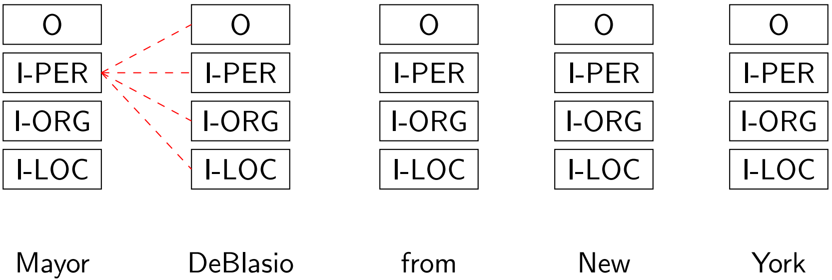
$$f(\mathbf{x}, c_{1:n}) = \sum_{i=1}^{n} \log \hat{y}(c_{i-1})_{c_i}$$

- Can extend these ideas to higher-order models.

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

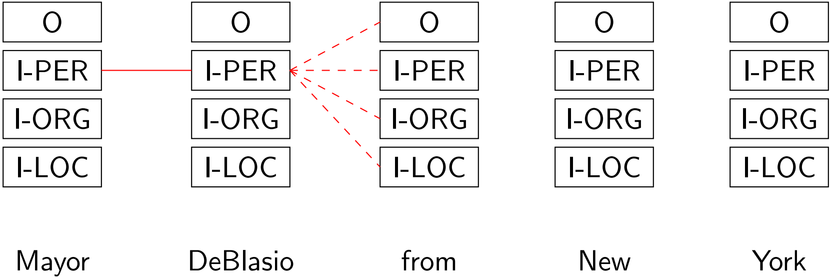
Review: Greedy Search

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



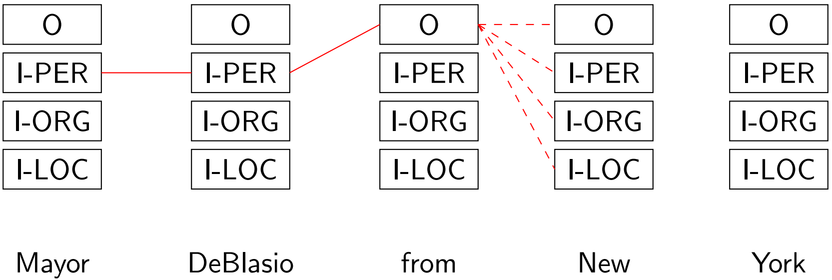
Review: Greedy Search

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



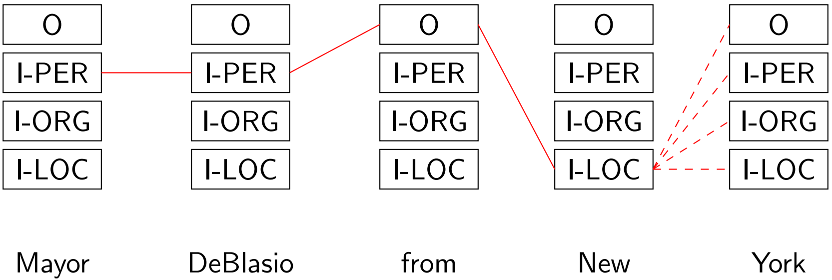
Review: Greedy Search

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



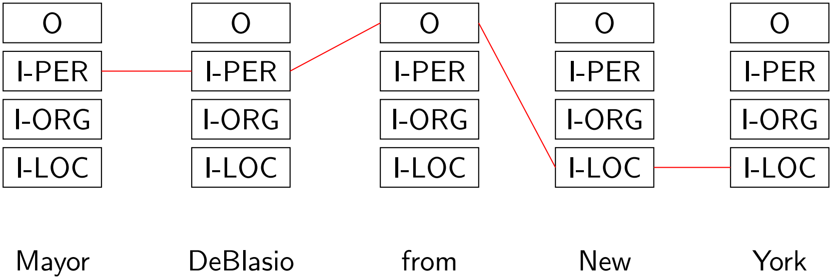
Review: Greedy Search

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



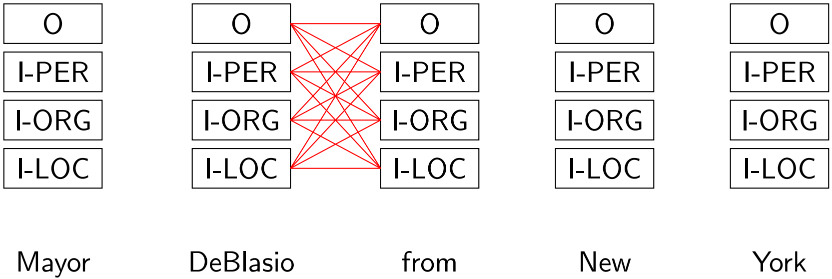
Review: Greedy Search

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



Review: 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 $i - 1$



Review: Viterbi Algorithm

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**

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

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

- Time Complexity?
- Space Complexity?

Quiz: Reverse it

Each of the algorithms goes left-to-right (forward) when producing a sequence. Sometimes you can derive right-to-left versions of these algorithms. Consider right-to-left cases of the following. Which are possible to run, how does the algorithm change, and do you get the same solutions?

1. Right-to-left greedy search on a Markov model.
2. Right-to-left Viterbi search on a Markov model.
3. Right-to-left greedy search on an RNN model

Answer: R-to-L Greedy Search

- In general, may not work.
- May require computing and renormalizing one step in the past $(O(|\mathcal{C}|^2))$
- Likely gives a different solution than forward greedy.

procedure $\textsc{GreedySearch}$

$s = 0$ ▷ Running score

$c_{n+1} = \langle / s \rangle$ ▷ Final Symbol

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

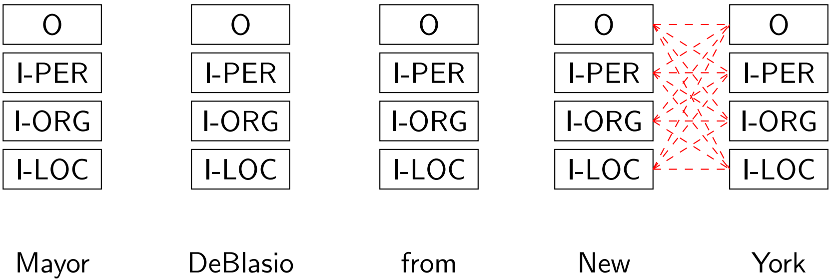
$$c_i \leftarrow \arg\max_{c_i'} \hat{y}(c_i')_{c_{i+1}}$$

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

return c, s

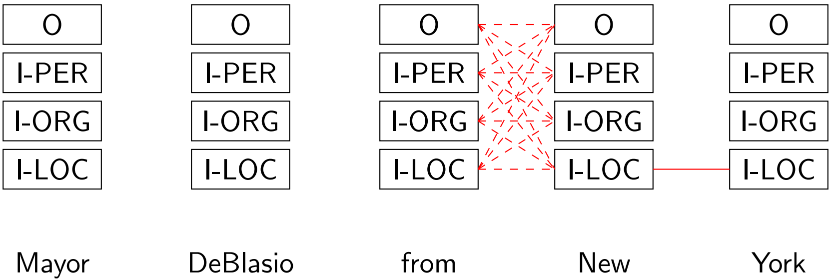
Answer: R-to-L Greedy Search (worst-case)

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



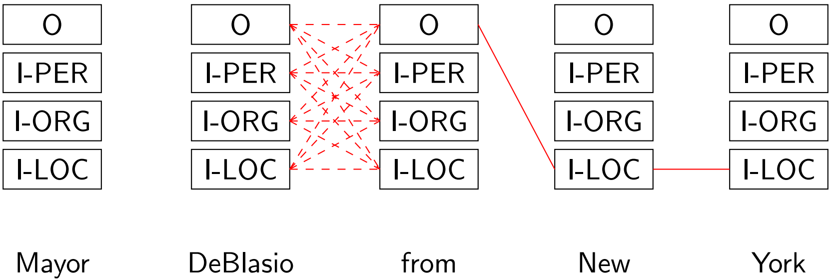
Answer: R-to-L Greedy Search (worst-case)

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



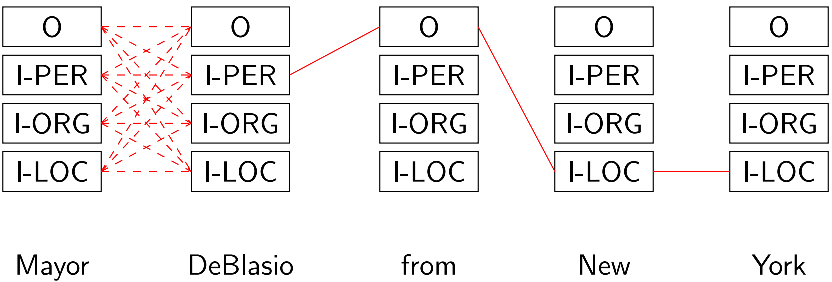
Answer: R-to-L Greedy Search (worst-case)

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



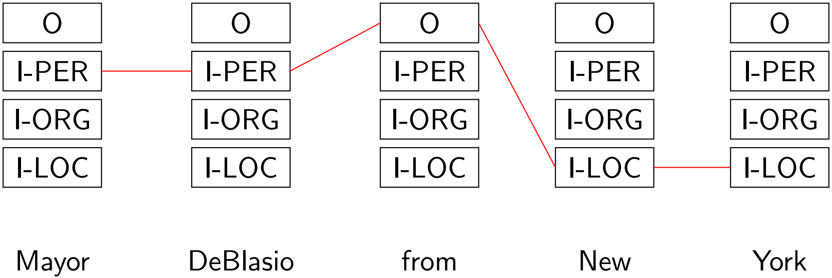
Answer: R-to-L Greedy Search (worst-case)

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



Answer: R-to-L Greedy Search (worst-case)

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



Answer: Backward Viterbi

- Same speed, same results.
- Similar inductive rule applies.
- Construct sequences starting at the end.

procedure $\textsc{BackwardViterbi}$

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

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

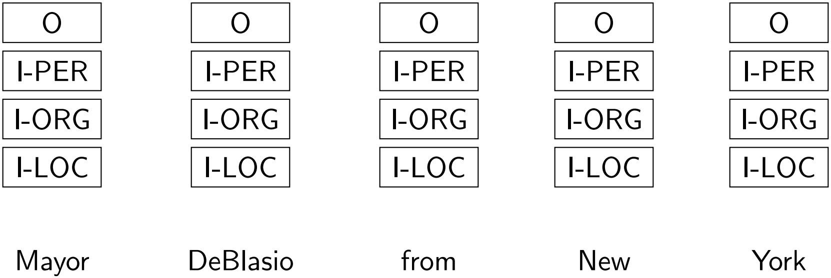
for $i = n$ to 1 **do**

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

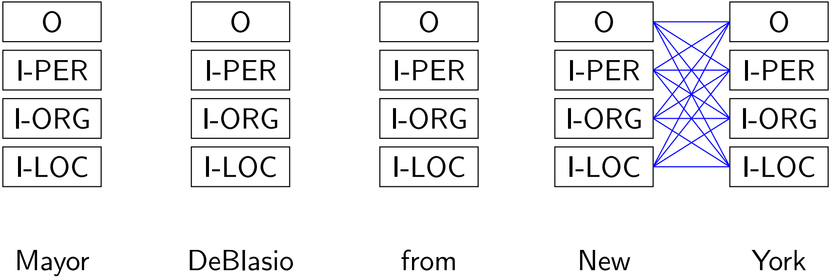
$\pi[i, c_i] = \max_{c'_{i+1}} \pi[i+1, c'_{i+1}] + \log \hat{y}(c_i)_{c'_{i+1}}$

return $\max_{c_1 \in \mathcal{C}} \pi[1, c_1]$

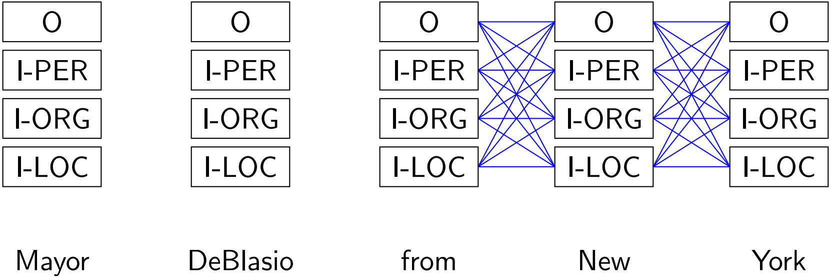
Backward Viterbi



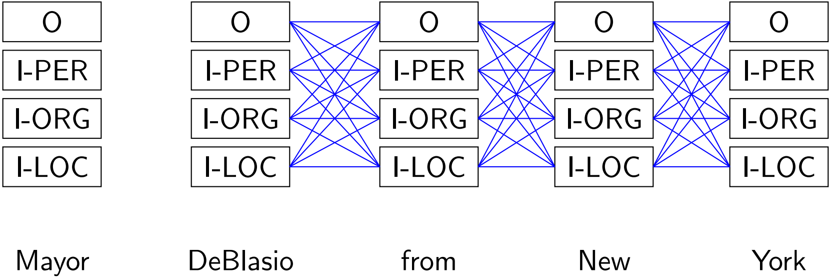
Backward Viterbi



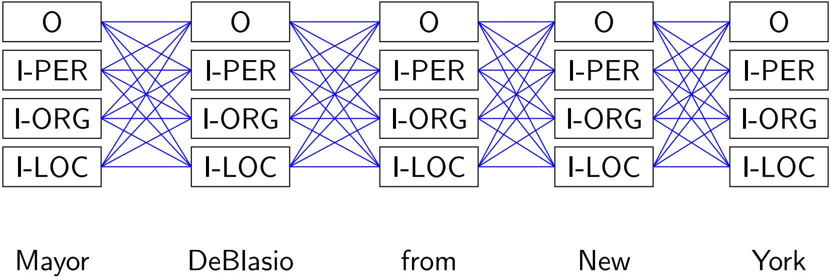
Backward Viterbi



Backward Viterbi



Backward Viterbi



Answer: Right-to-Left Greedy RNN

- Does not work.
- Have suffix $c_n \dots c_i$ but RNN is prefix-function $\hat{\mathbf{y}}(c_1, \dots c_i)$.
- Backward greedy would require enumerating all prefixes.

Alternative Solution: Backward RNN

- Can train RNN in the reverse direction.
- RNN suffix transducer trained on $\hat{\mathbf{y}}(c_n, \ldots c_i)$.
- However note this is a different model.
- Even if you had exact search, this may yield a different output.

Today's Lecture

- More Dynamic Programming
  - Max-Marginals
  - Forward-Backward
  - Probabilistic Marginals
  - Pruning
- Next Lecture: Conditional Random Fields

Question 1: Tag Fill-In

Assume we are given $c_{1:i-1}$ and $c_{i+1:n}$,

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

What is the best completion, i.e.

$$\arg\max_{c'_i} f(\mathbf{x}, c_{1:i-1} : c'_i : c_{i+1:n})$$

Question 1: Tag Fill-In

Assume we are given $c_{1:i-1}$ and $c_{i+1:n}$,

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

What is the best completion, i.e.

$$\arg\max_{c'_i} f\left(\mathbf{x}, c_{1:i-1} : c'_i : c_{i+1:n}\right)$$

Answer

- Markov model. Score involving c_i are local (c_{i-1} and c_{i+1}).
- Can solve by looking one-step forward and backward

$$\arg\max_{c'_i} f(\mathbf{x}, c_{1:n}) = \arg\max_{c'_i} \log \hat{y}(c_{i-1})_{c'_i} + \log \hat{y}(c'_i)_{c_{i+1}}$$

- Can be solved in $O(|\mathcal{C}|)$ or $O(|\mathcal{C}|^2)$ depending on model.

Exercise 2: Sequence fill-in

Assume we **are not** given $c_{1:i-1}$ and $c_{i+1:n}$,

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

What is the score of the max sequence through each tag, i.e.

$$M(i, c'_i) = \max_{c_{1:i-1}, c_{i+1:n}} f(\mathbf{x}, c_{1:i-1} : c'_i : c_{i+1:n})$$

Answer

- Markov property. Two prefix and suffix score can are independent.
- $$\begin{aligned} M(i, c'_i) &= \max_{c_{1:i-1}:c_{i+1:n}} f(\mathbf{x}, c_{1:i-1} : c'_i : c_{i+1:n}) \\ &= \max_{c_{1:i-1}} \log \hat{y}(c'_{i-1})_{c'_i} + \sum_{j=1}^{i} \log \hat{y}(c_{j-1})_{c_j} \\ &+ \max_{c_{i+1:n}} \log \hat{y}(c'_i)_{c_{i+1}} + \sum_{j=i+1}^{n} \log \hat{y}(c_j)_{c_{j+1}} \end{aligned}$$

Viterbi Forward-Backward

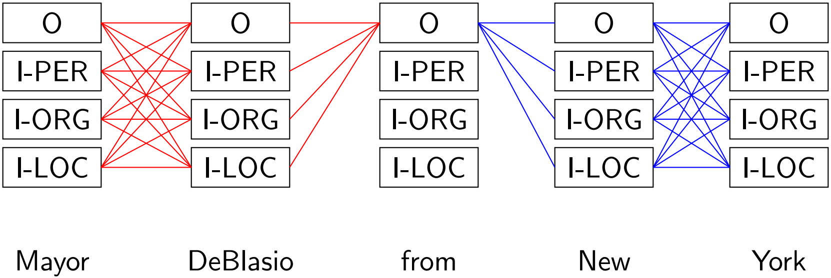
$$
\begin{aligned}
& \max_{c_{1:i-1}} \log \hat{y}(c'_{i-1})_{c'_i} + \sum_{j=1}^{i} \log \hat{y}(c_{j-1})_{c_j} \\
+ \quad & \max_{c_{i+1:n}} \log \hat{y}(c'_i)_{c_{i+1}} + \sum_{j=i+1}^{n} \log \hat{y}(c_j)_{c_{j+1}}
\end{aligned}
$$

- Forward Viterbi scores (max prefix)

$$
\pi^{\alpha}[i, c'_i] = \max_{c_{1:i-1}} \log \hat{y}(c'_{i-1})_{c'_i} + \sum_{j=1}^{i} \log \hat{y}(c_{j-1})_{c_j}
$$

- Backward Viterbi scores (max suffix)

$$
\pi^{\beta}[i, c'_i] = \max_{c_{i+1:n}} \log \hat{y}(c'_i)_{c_{i+1}} + \sum_{j=i+1}^{n} \log \hat{y}(c_j)_{c_{j+1}}
$$


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

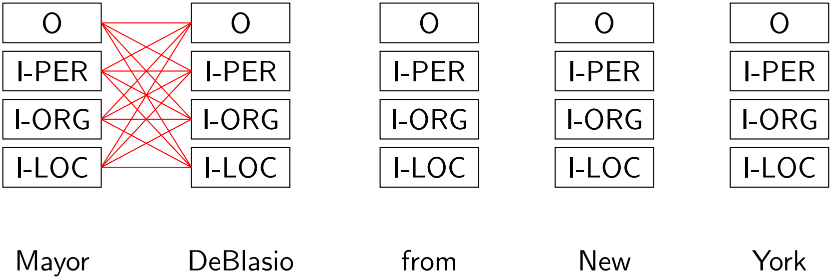
Computing All Max-Marginals

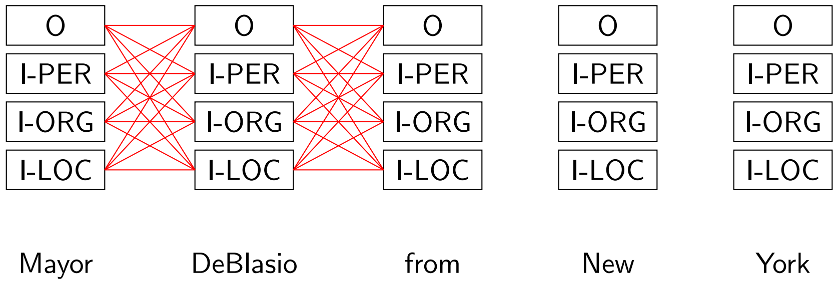
$$\arg\max_{c_i} \max_{c_{1:i-1}:c_{i+1:n}} f(\mathbf{x}, c_{1:n})$$

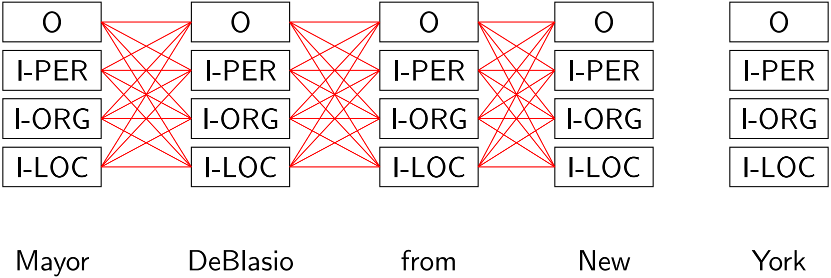
- Compute π^α using Viterbi forward
- Compute π^β using Viterbi backward
- Compute the argmax

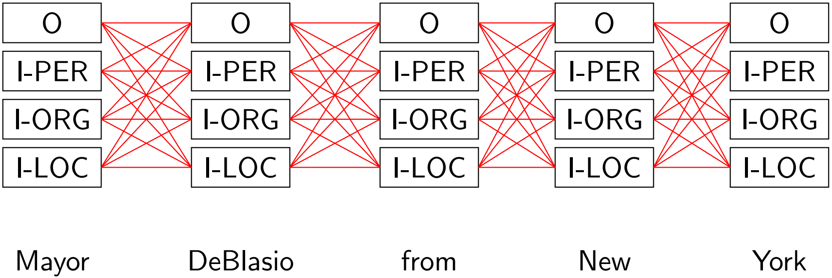
$$M(i, c_i') = \pi^\alpha[i, c_i'] + \pi^\beta[i, c_i']$$

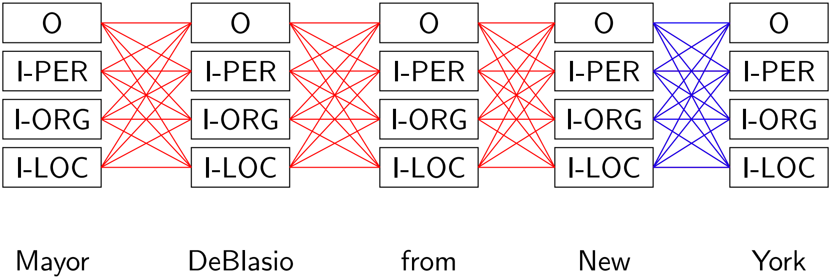
- Time complexity?
- Space complexity?

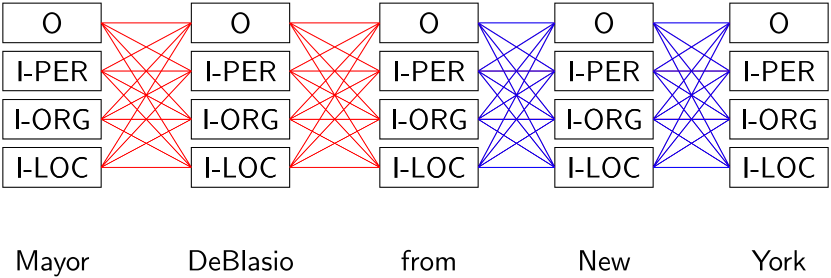

O
I-PER
I-ORG
I-LOC
Mayor
DeBlasio
from
New
York

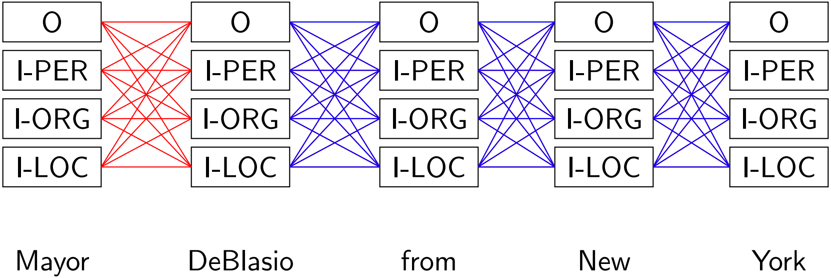

O
I-PER
I-ORG
I-LOC
Mayor
DeBlasio
from
New
York

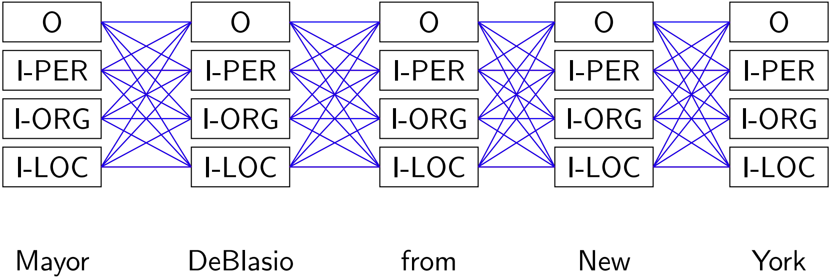

O
I-PER
I-ORG
I-LOC
Mayor
O
I-PER
I-ORG
I-LOC
DeBlasio
O
I-PER
I-ORG
I-LOC
from
O
I-PER
I-ORG
I-LOC
New
O
I-PER
I-ORG
I-LOC
York


O
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

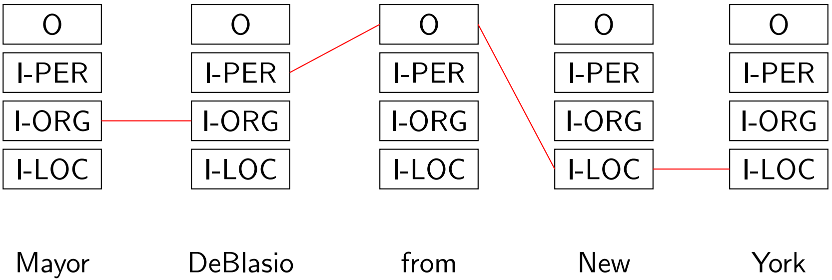
Max-Marginals with Backpointers

- Can also include forward and backward backpointers
- Get best sequence through any tag in $O(n|\mathcal{C}|^2)$.

Max-Marginal Property

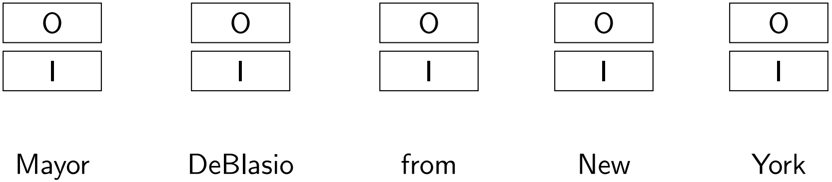
If c_i^* is part of highest-scoring sequence then max-marginal at $M(i, c_i^*)$ is $\max_{c_{1:n}} f(\mathbf{x}, c_{1:n})$ and by definition is at least as large as any other max-marginal $M(j, c_j)$ for all j, c_j.

Pruning by Bounding (Sketch)



- First run greedy over full problem.
- Score $s^{(greedy)}$ is less than optimal.

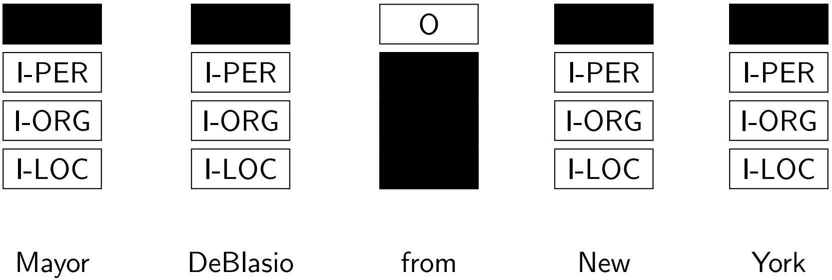
Pruning by Bounding (Sketch)



- Each edge is the max of the edges in the full lattice.
- Can compute max-marginals M in much lkess time.

Pruning by Bounding (Sketch)

- For all j, c_j, if $M(j, c_j) < s^{greedy}$ can prune.
- Running Viterbi over pruned lattice is provably optimal.



Contents

Marginals

Let us return to the case of discriminative probabilistic models.

- Model of

$$p(\mathbf{y} = \delta(c_{1:n})|\mathbf{x})$$

-

Exercise 3: Smoothing

Assume we are given $c_{1:i-1}$ and $c_{i+1:n}$,

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

What is the probability of $\mathbf{y}_i$, i.e.

$$p(\mathbf{y}_i = \delta(c_i) | \mathbf{y}_{1:i-1} = \delta(c_{1:i-1}), \mathbf{y}_{i+1:n} = \delta(c_{i+1:n}), \mathbf{x})$$

Answer

- Same idea. Score involving c_i are local ($i-1$ and $i+1$).
- Can compute "smoothing" distribution from local information

$$
\begin{aligned}
p(\mathbf{y}_i = \delta(c_i) | \mathbf{y}_{1:i-1}, \mathbf{y}_{i+1:n}, \mathbf{x}) &\propto p(\mathbf{y}_i | \mathbf{y}_{i-1}) p(\mathbf{y}_{i+1} | \mathbf{y}_i) \\
&= \hat{y}(c_{i-1})_{c'_i} \hat{y}(c'_i)_{c_{i+1}}
\end{aligned}
$$

Exercise 4

Assume we **are not** given $c_{1:i-1}$ and $c_{i+1:n}$.

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

What is the best completed sequence, i.e.

$$p(\mathbf{y}_i = \delta(c_i)|\mathbf{x})$$

Answer: Marginalization

- Similar idea. Score involving c_i are local ($i-1$ and $i+1$).

$$
\begin{aligned}
p(\mathbf{y}_i = \delta(c_i')|\mathbf{x}) &= \sum_{c_{1:i-1}:c_{i+1:n}} p(\mathbf{y}_i = \delta(c_i'), \mathbf{y}_{1:i-1,i+1:n}|\mathbf{x}) \\
&= \sum_{c_{1:i-1}} p(\mathbf{y}_{1:i-1}|\mathbf{x}) p(\mathbf{y}_i = \delta(c_i')|\mathbf{y}_{i-1}, \mathbf{x}) \\
&\times \sum_{c_{i+1:n}} p(\mathbf{y}_{i+1}|\mathbf{y}_i =, \mathbf{x}) p(\mathbf{y}_{i+1:n}|\mathbf{x}) \\
&= \sum_{c_{1:i-1}} \hat{y}(c_{i-1})_{c_i'} \prod_{j=1}^{i-1} \hat{y}(c_{j-1})_{c_j} \\
&\times \sum_{c_{i+1:n}} \hat{y}(c_i')_{c_{i+1}} \prod_{j=i+1}^{n} \hat{y}(c_j)_{c_{j+1}}
\end{aligned}
$$

Answer: Marginalization

- Similar idea. Score involving c_i are local ($i-1$ and $i+1$).

$$
\begin{aligned}
p(\mathbf{y}_i = \delta(c_i')|\mathbf{x}) &= \sum_{c_{1:i-1}:c_{i+1:n}} p(\mathbf{y}_i = \delta(c_i'), \mathbf{y}_{1:i-1,i+1:n}|\mathbf{x}) \\
&= \sum_{c_{1:i-1}} p(\mathbf{y}_{1:i-1}|\mathbf{x}) p(\mathbf{y}_i = \delta(c_i')|\mathbf{y}_{i-1}, \mathbf{x}) \\
&\times \sum_{c_{i+1:n}} p(\mathbf{y}_{i+1}|\mathbf{y}_i =, \mathbf{x}) p(\mathbf{y}_{i+1:n}|\mathbf{x}) \\
&= \sum_{c_{1:i-1}} \hat{y}(c_{i-1})_{c_i'} \prod_{j=1}^{i-1} \hat{y}(c_{j-1})_{c_j} \\
&\times \sum_{c_{i+1:n}} \hat{y}(c_i')_{c_{i+1}} \prod_{j=i+1}^{n} \hat{y}(c_j)_{c_{j+1}}
\end{aligned}
$$

Forward and Backward

- Forward scores (sum over prefixes)

$$\alpha[i, c_i'] = \sum_{c_{1:i-1}} \hat{y}(c_{i-1})_{c_i'} \prod_{j=1}^{i-1} \hat{y}(c_{j-1})_{c_j}$$

- Backward scores (sum over suffixes)

$$\beta[i, c_i'] = \sum_{c_{i+1:n}} \hat{y}(c_i')_{c_{i+1}} \prod_{j=i+1}^{n} \hat{y}(c_j)_{c_{j+1}}$$

Forward Algorithm

procedure FORWARD

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

$\alpha[0, \langle s \rangle] = 1$

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

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

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

return α

Backward Algorithm

procedure BACKWARD

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

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

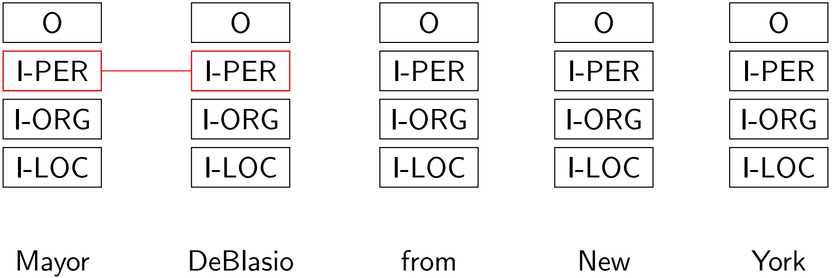
for $i = n$ to 1 **do**

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

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

return β

Exercise 5: Edge Marginals



What is the probability of using an edge, i.e.

$$p(\mathbf{y}_i = \delta(c'_i), \mathbf{y}_{i+1} = \delta(c'_{i+1}), |\mathbf{x})$$

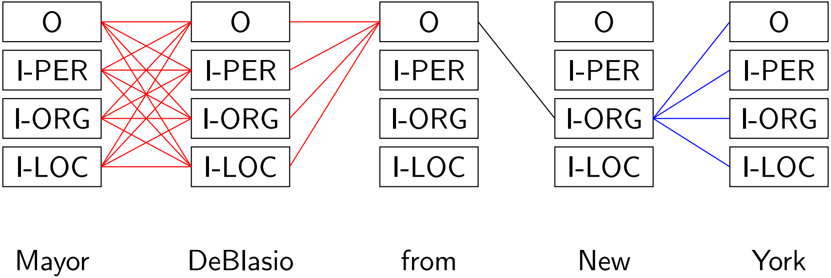
Edge Marginals

$$\hat{y}(c'_i)_{c'_{i+1}} \times \sum_{c_{1:i-1}} \hat{y}(c_{i-1})_{c'_i} \prod_{j=1}^{i-1} \hat{y}(c_{j-1})_{c_j}$$
$$\times \quad \sum_{c_{i+2:n}} \hat{y}(c'_{i+1})_{c_{i+1}} \prod_{j=i+1}^{n} \hat{y}(c_j)_{c_{j+1}}$$

- Compute α using Forward
- Compute β using Backward
- Multiply in the edge

$$\hat{y}(c'_i)_{c'_{i+1}} \alpha[i, c'_i] \times \beta[i+1, c'_{i+1}]$$

Edge Marginal



Marginals versus Max-Marginals

- Max-Marginals: Most-likely sequence through decision
- Marginals: Sum of sequences through decision.

- Possibly very different values.
- Edge with highest marginal may not be in best sequence.

Edge Marginal Decoding

- For all i

$$p(y_i = \delta(c_i), y_{i+1} = \delta(c_{i+1}) | \mathbf{x})$$

- But this is a Markov model!
- Replace lattice with edge marginals

$$f(\mathbf{x}, c_{1:n}) = \sum_i \log p(y_i = \delta(c_i), y_{i+1} = \delta(c_{i+1}) | \mathbf{x})$$

- Posterior decoding.

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