Sequential Supervised Learning

Many Application Problems Require Sequential Learning

- Part-of-speech Tagging
- Information Extraction from the Web
- Text-to-Speech Mapping

Part-of-Speech Tagging

- Given an English sentence, can we assign a part of speech to each word?

- "Do you want fries with that?"
- <verb pron verb noun prep pron>

Information Extraction from the Web

<dl><dt>Srinivasan Seshan (Carnegie Mellon University) <dt><i>Making Virtual Worlds Real</i><dt>Tuesday, June 4, 2002<dd>2:00 PM , 322 Sieg<dd>Research Seminar

* * * name name * * affiliation affiliation affiliation * * * * title title title title * * * date date date date * time time * location location * event-type event-type

Text-to-Speech Mapping

- “photograph” `=> /f-Ot@graf-/`

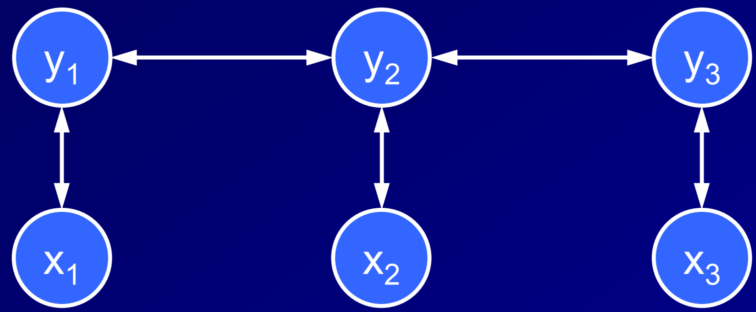
Sequential Supervised Learning (SSL)

- Given: A set of training examples of the form $(\mathbf{X}_i, \mathbf{Y}_i)$, where
  $\mathbf{X}_i = \langle x_{i,1}, \ldots, x_{i,T_i} \rangle$ and
  $\mathbf{Y}_i = \langle y_{i,1}, \ldots, y_{i,T_i} \rangle$ are sequences of length T_i
- Find: A function f for predicting new sequences: $\mathbf{Y} = f(\mathbf{X})$.

Examples of Sequential Supervised Learning

Domain	Input $\mathbf{X}_i$	Output $\mathbf{Y}_i$
Part-of-speech Tagging	sequence of words	sequence of parts of speech
Information Extraction	sequence of tokens	sequence of field labels {name, …}
Test-to-speech Mapping	sequence of letters	sequence phonemes

Two Kinds of Relationships



- "Vertical" relationship between the x_t's and y_t's
  - Example: "Friday" is usually a "date"
- "Horizontal" relationships among the y_t's
  - Example: "name" is usually followed by "affiliation"
- SSL can (and should) exploit both kinds of information

Existing Methods

- Hacks
  - Sliding windows
  - Recurrent sliding windows
- Hidden Markov models
  - joint distribution: $P(X,Y)$
- Conditional Random Fields
  - conditional distribution: $P(Y|X)$
- Discriminant Methods: HM-SVMs, MMMs, voted perceptrons
  - discriminant function: $f(Y; X)$

Sliding Windows

___	Do	you	want	fries	with	that	___

___	Do	you	→	verb
Do	you	want	→	pron
you	want	fries	→	verb
want	fries	with	→	noun
fries	with	that	→	prep
with	that	___	→	pron

Properties of Sliding Windows

- Converts SSL to ordinary supervised learning
- Only captures the relationship between (part of) X and y_t. Does not explicitly model relations among the y_t's
- Assumes each window is independent

Recurrent Sliding Windows

___	Do	you	want	fries	with	that	___

___	Do	you	___	→	verb
Do	you	want	verb	→	pron
you	want	fries	pron	→	verb
want	fries	with	verb	→	noun
fries	with	that	noun	→	prep
with	that	___	prep	→	pron

Recurrent Sliding Windows

- Key Idea: Include y_t as input feature when computing y_{t+1}.
- During training:
  - Use the correct value of y_t
  - Or train iteratively (especially recurrent neural networks)
- During evaluation:
  - Use the predicted value of y_t

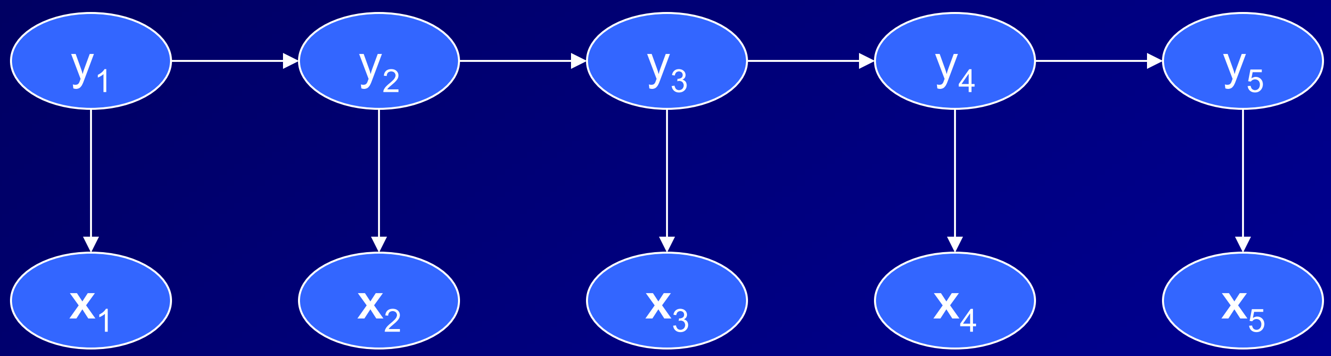
Properties of Recurrent Sliding Windows

- Captures relationship among the y's, but only in one direction!
- Results on text-to-speech:

Method	Direction	Words	Letters
sliding window	none	12.5%	69.6%
recurrent s. w.	left-right	17.0%	67.9%
recurrent s. w.	right-left	24.4%	74.2%

Hidden Markov Models

- Generalization of Naïve Bayes to SSL

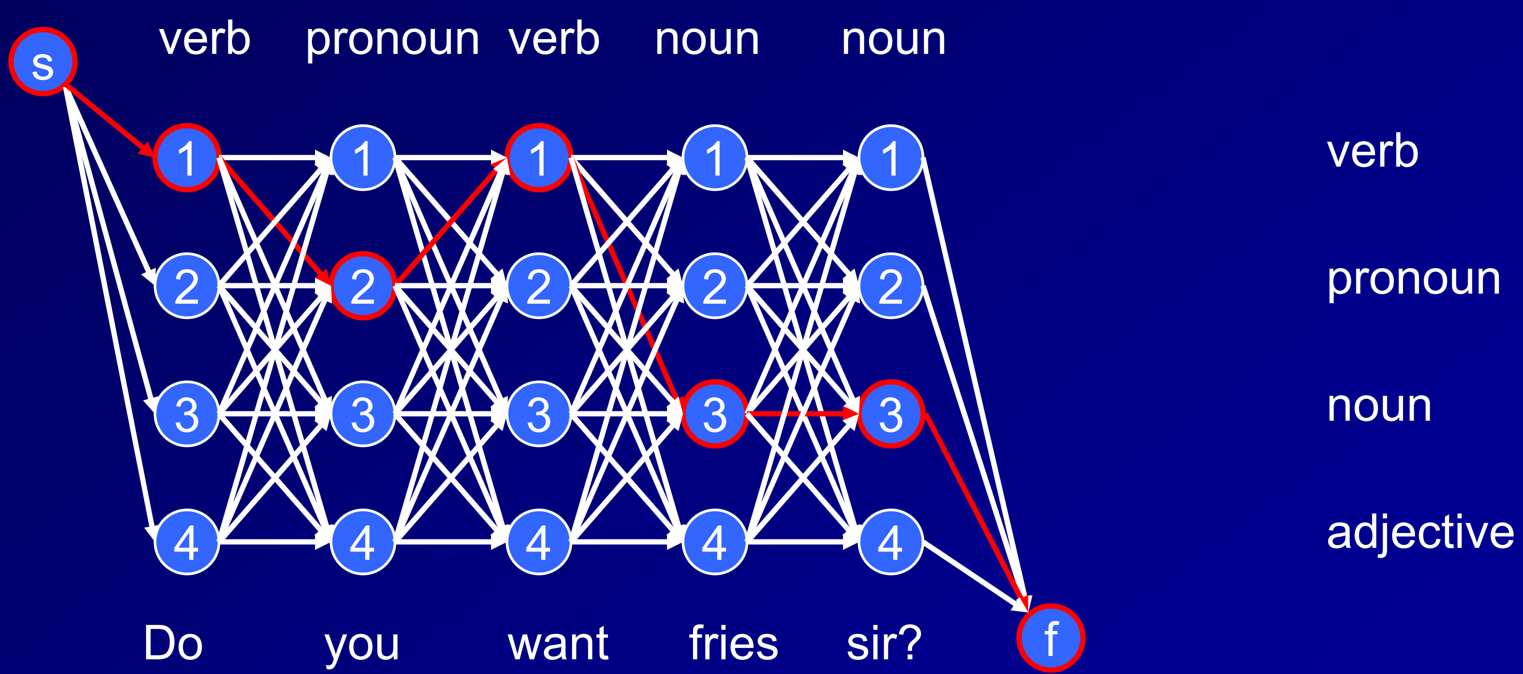


- $P(y_1)$
- $P(y_t \mid y_{t-1})$ assumed the same for all t
- $P(\mathbf{x}_t \mid y_t) = P(x_{t,1} \mid y_t) \cdot P(x_{t,2} \mid y_t) \cdots P(x_{t,n}, y_t)$ assumed the same for all t

Making Predictions with HMMs

- Two possible goals:
  - $\mathrm{argmax}_Y P(Y|X)$
    - find the most likely <u>sequence</u> of labels Y given the input sequence X
  - $\mathrm{argmax}_{y_t} P(y_t \mid X)$ forall t
    - find the most likely label y_t at each time t given the entire input sequence X

Finding Most Likely Label Sequence: The Trellis

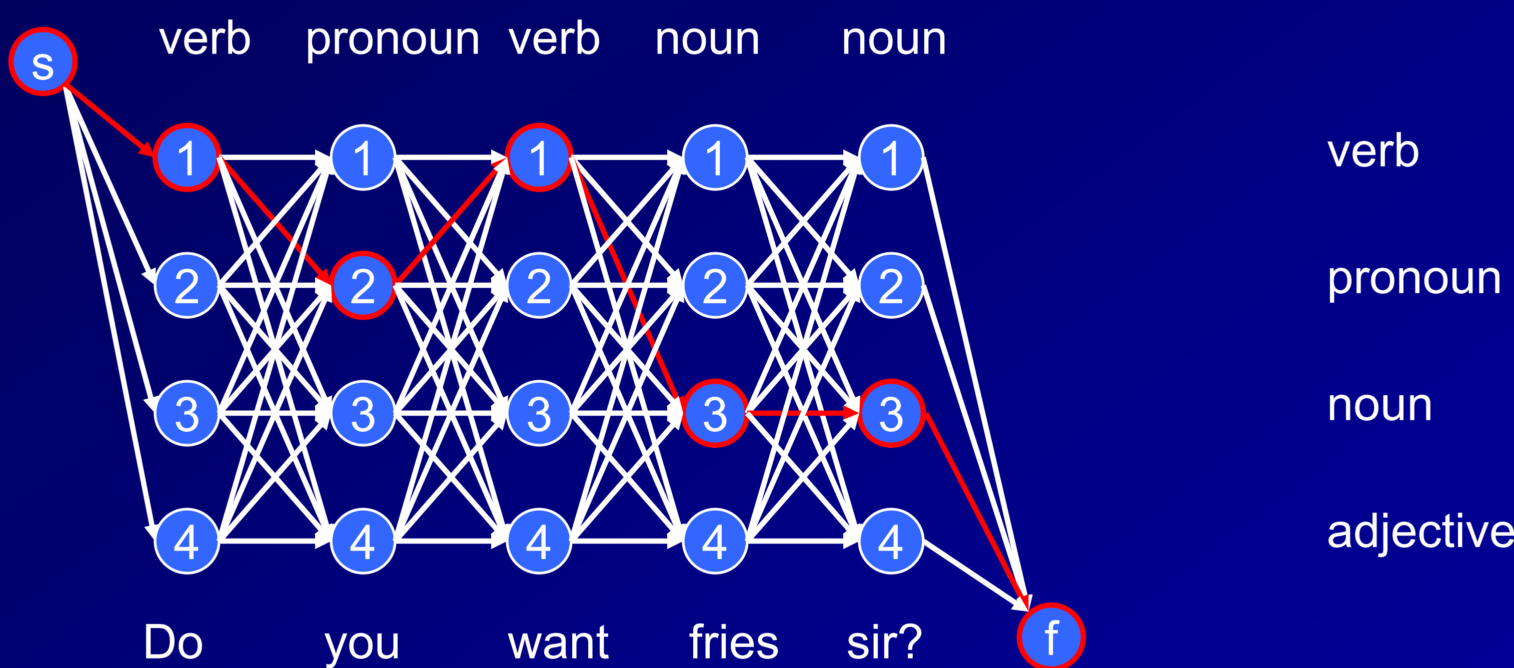


Every label sequence corresponds to a path through the trellis graph.

The probability of a label sequence is proportional to

$$P(y_1) \cdot P(x_1|y_1) \cdot P(y_2|y_1) \cdot P(x_2|y_2) \cdots P(y_T \mid y_{T-1}) \cdot P(x_T \mid y_T)$$

Converting to Shortest Path Problem

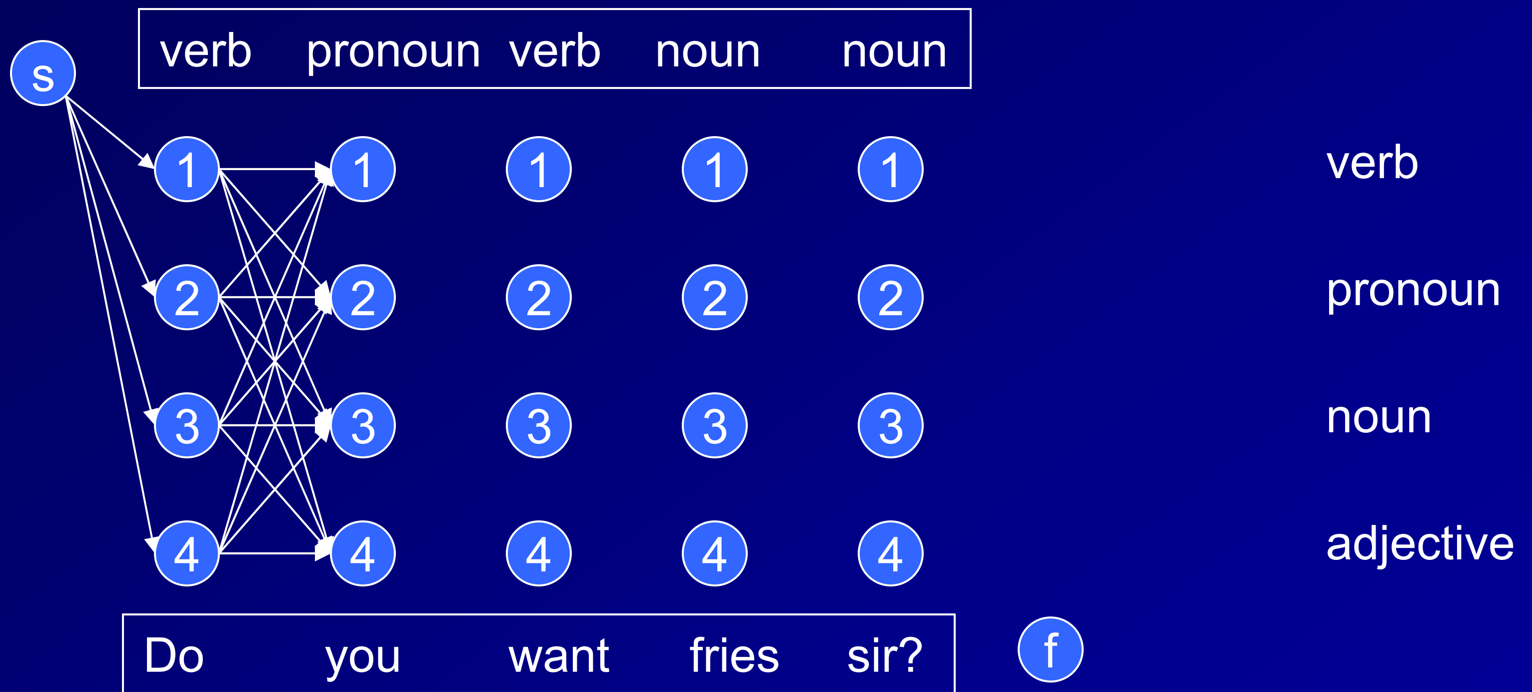


$\max_{y_1,\ldots,y_T} P(y_1) \cdot P(x_1|y_1) \cdot P(y_2|y_1) \cdot P(x_2|y_2) \cdots P(y_T \mid y_{T-1}) \cdot P(x_T \mid y_T) =$

$\min_{y_1,\ldots,y_T} -\log [P(y_1) \cdot P(x_1|y_1)] + -\log [P(y_2|y_1) \cdot P(x_2|y_2)] + \cdots + -\log [P(y_T \mid y_{T-1}) \cdot P(x_T \mid y_T)]$

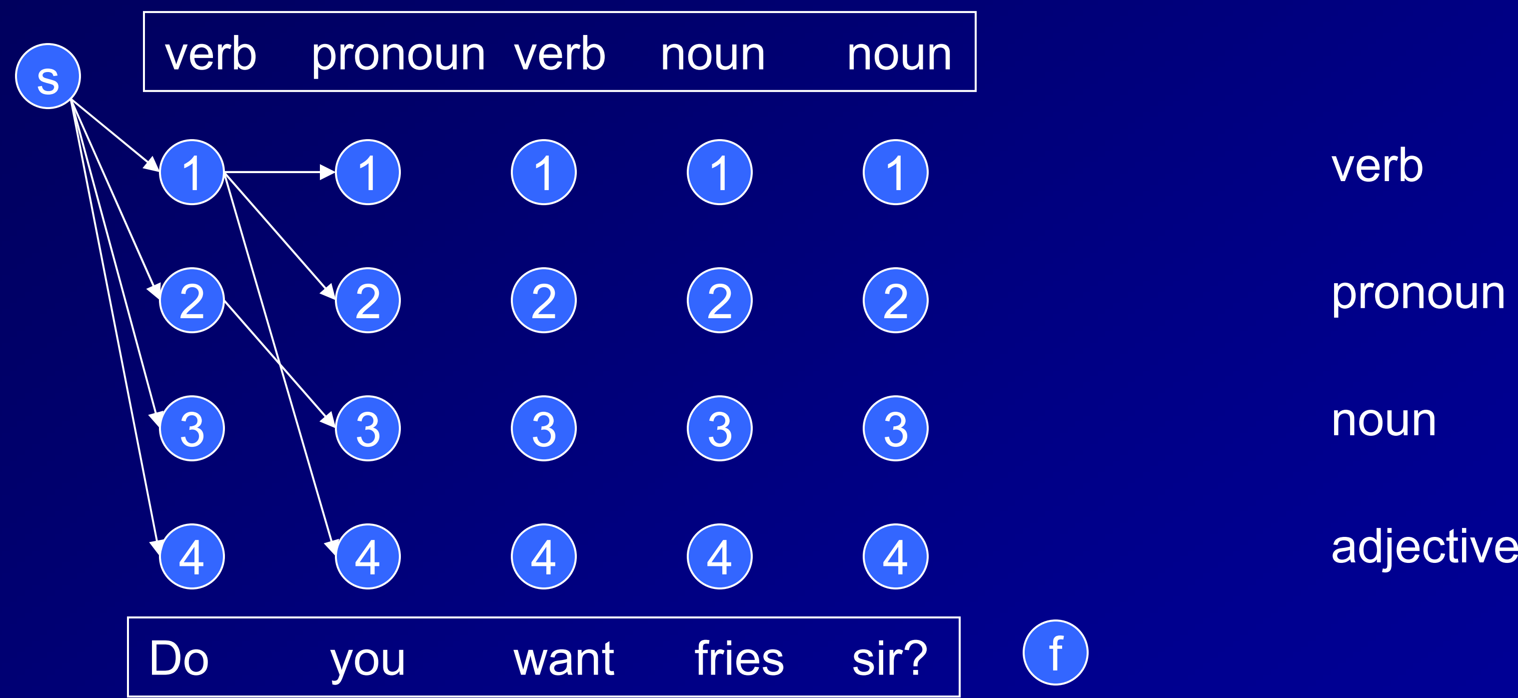
shortest path through graph. edge cost $= -\log [P(y_t|y_{t-1}) \cdot P(x_t|y_t)]$

Finding Most Likely Label Sequence: The Viterbi Algorithm



Step t of the Viterbi algorithm computes the possible successors of state y_{t-1} and computes the total path length for each edge

Finding Most Likely Label Sequence: The Viterbi Algorithm

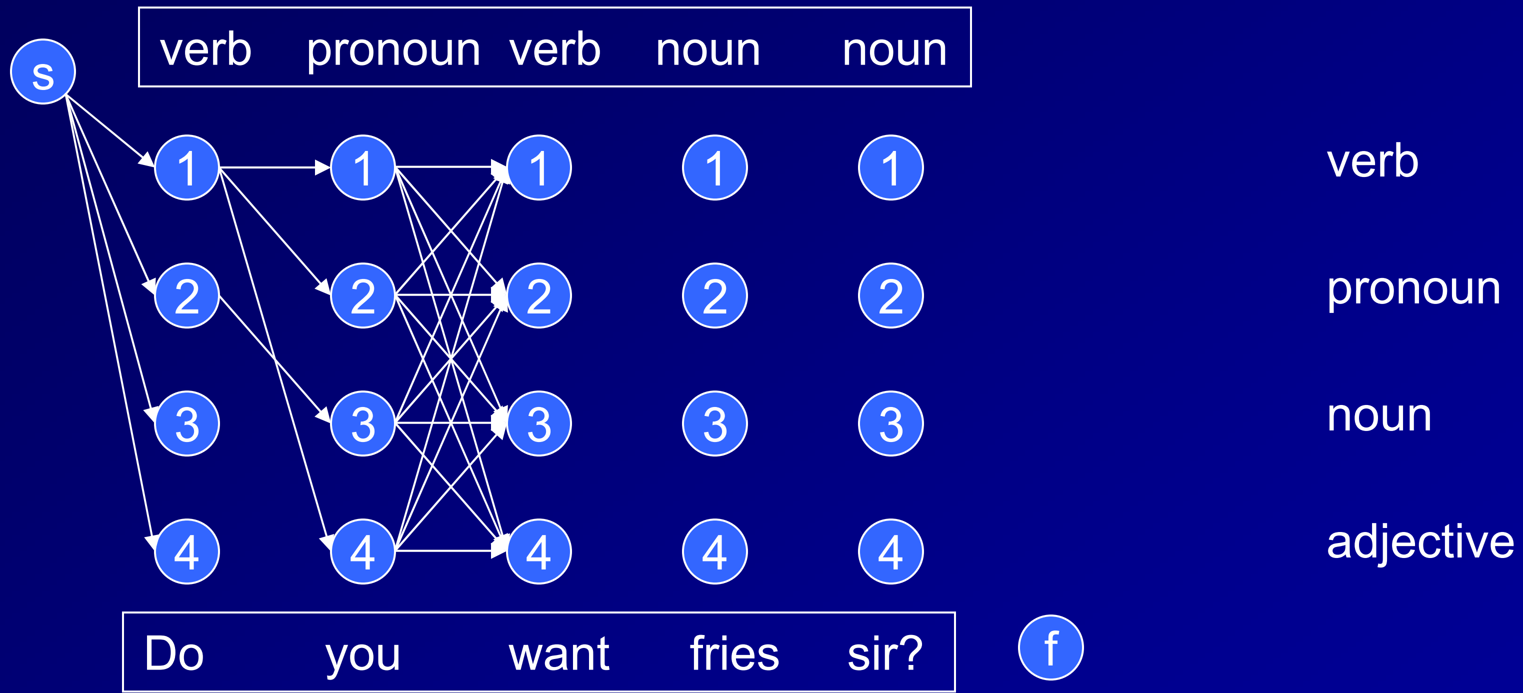


Each node y_t=k stores the cost μ of the shortest path that reaches it from s and the predecessor class y_{t-1} = k' that achieves this cost

$$k' = \arg\min_{y_{t-1}} -\log [P(y_t \mid y_{t-1}) \cdot P(x_t \mid y_t)] + \mu(y_{t-1})$$

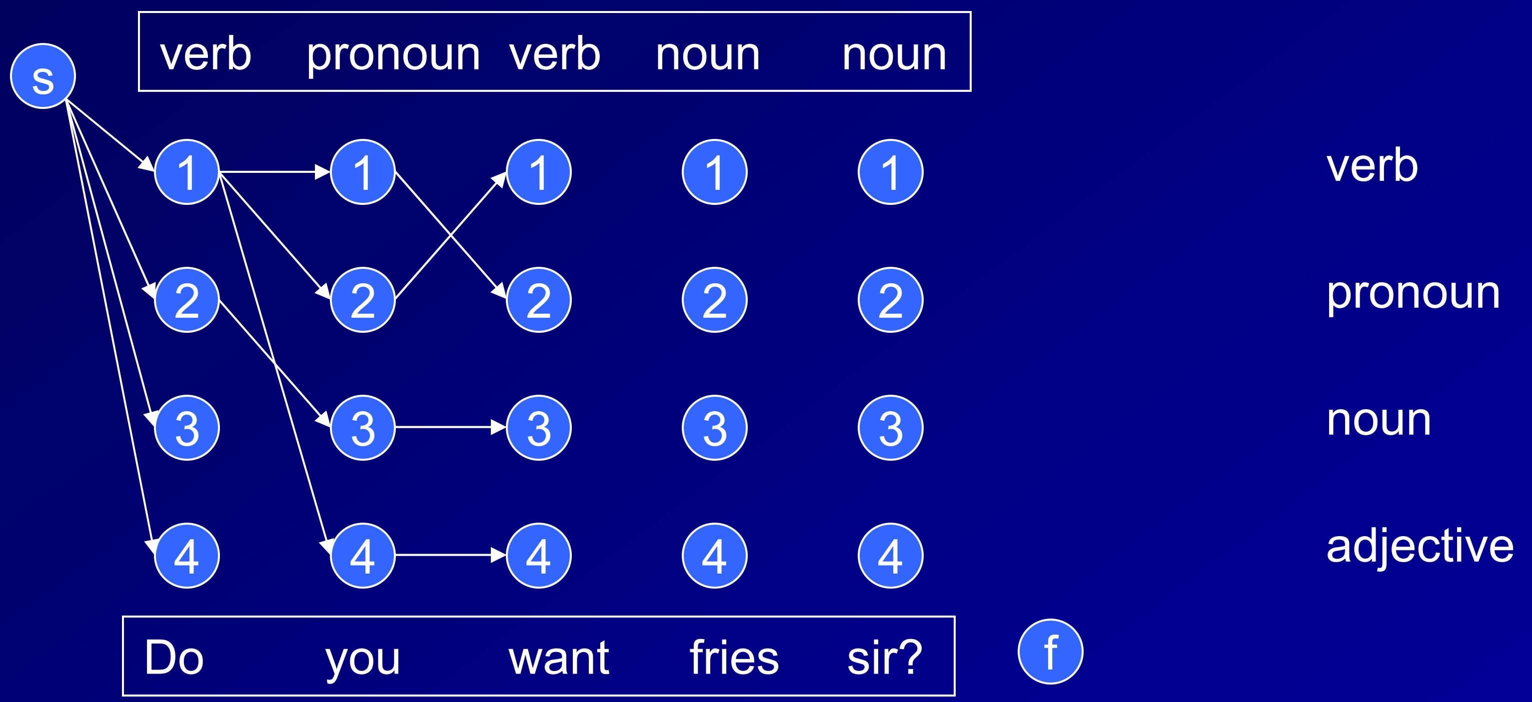
$$\mu(k) = \min_{y_{t-1}} -\log [P(y_t \mid y_{t-1}) \cdot P(x_t \mid y_t)] + \mu(y_{t-1})$$

Finding Most Likely Label Sequence: The Viterbi Algorithm



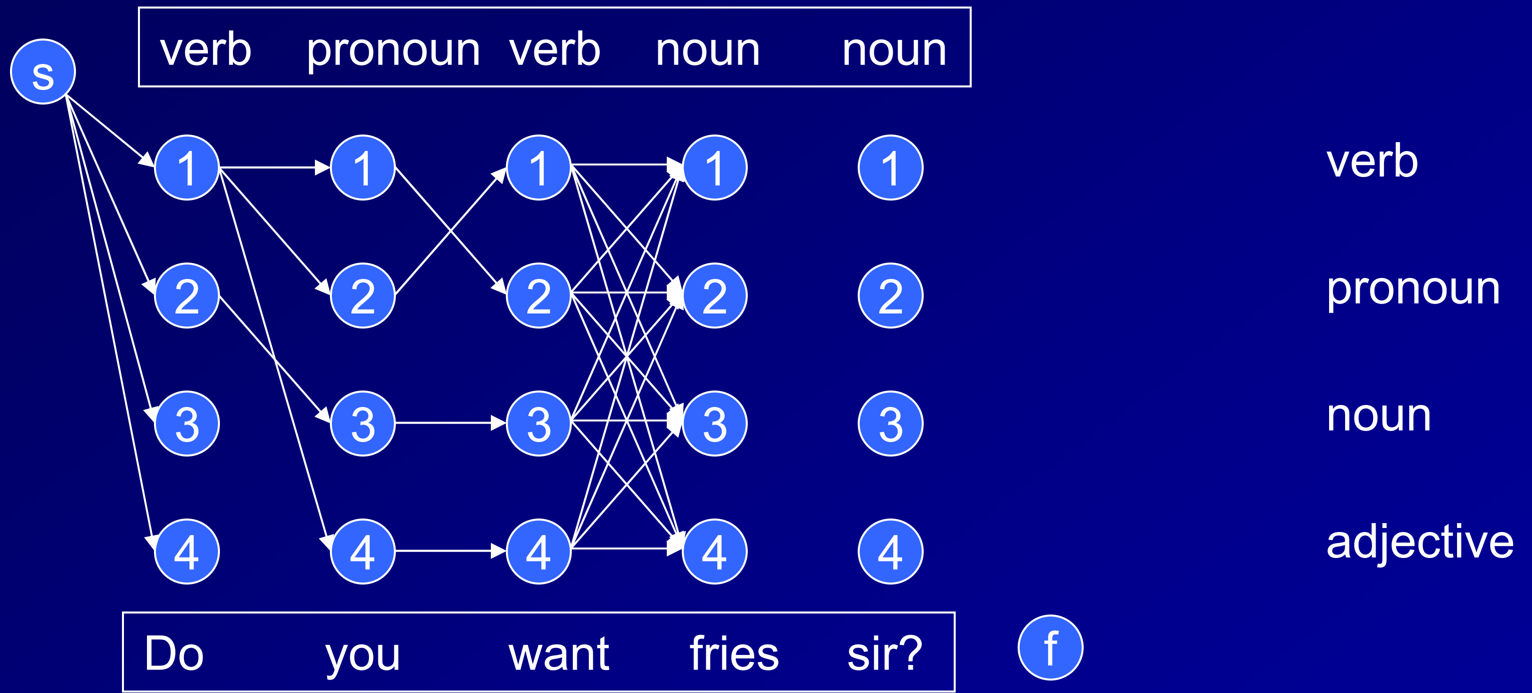
Compute Successors…

Finding Most Likely Label Sequence: The Viterbi Algorithm



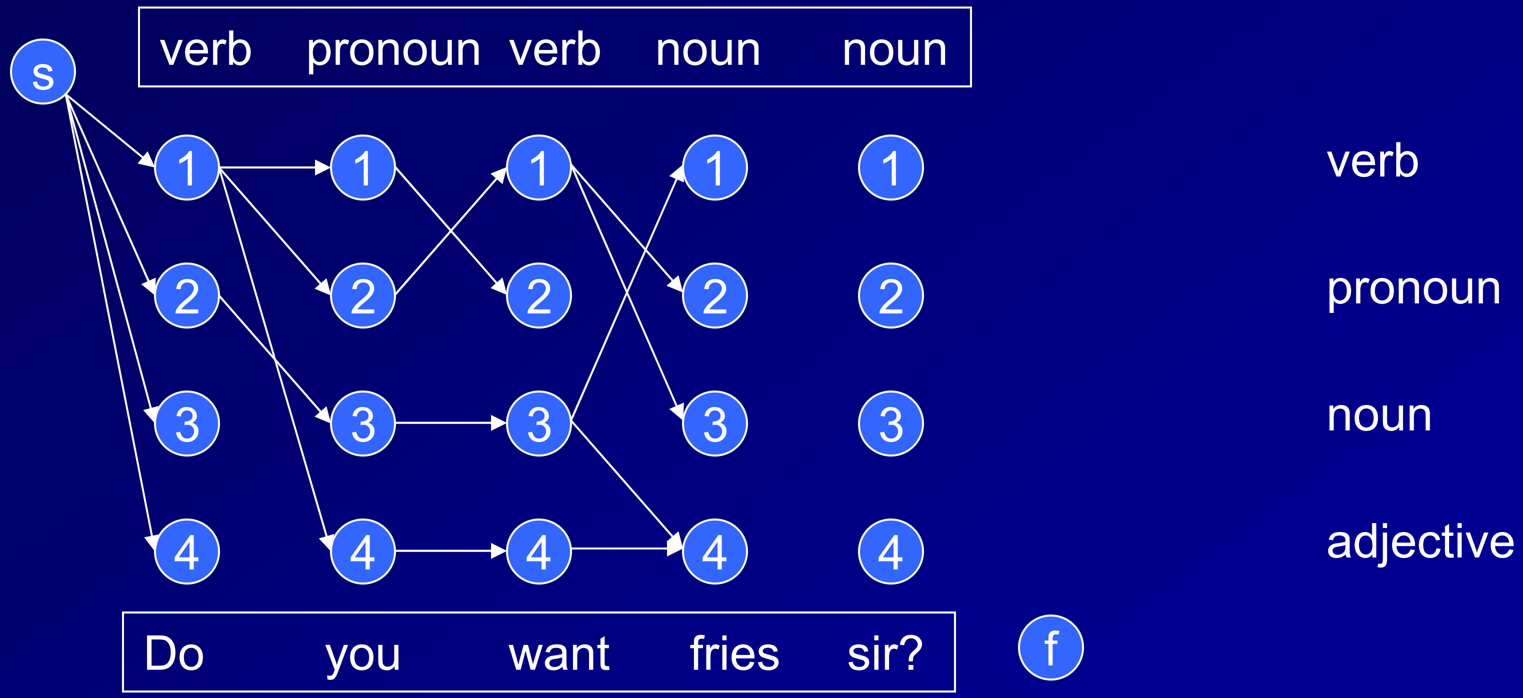
Compute and store shortest incoming arc at each node

Finding Most Likely Label Sequence: The Viterbi Algorithm



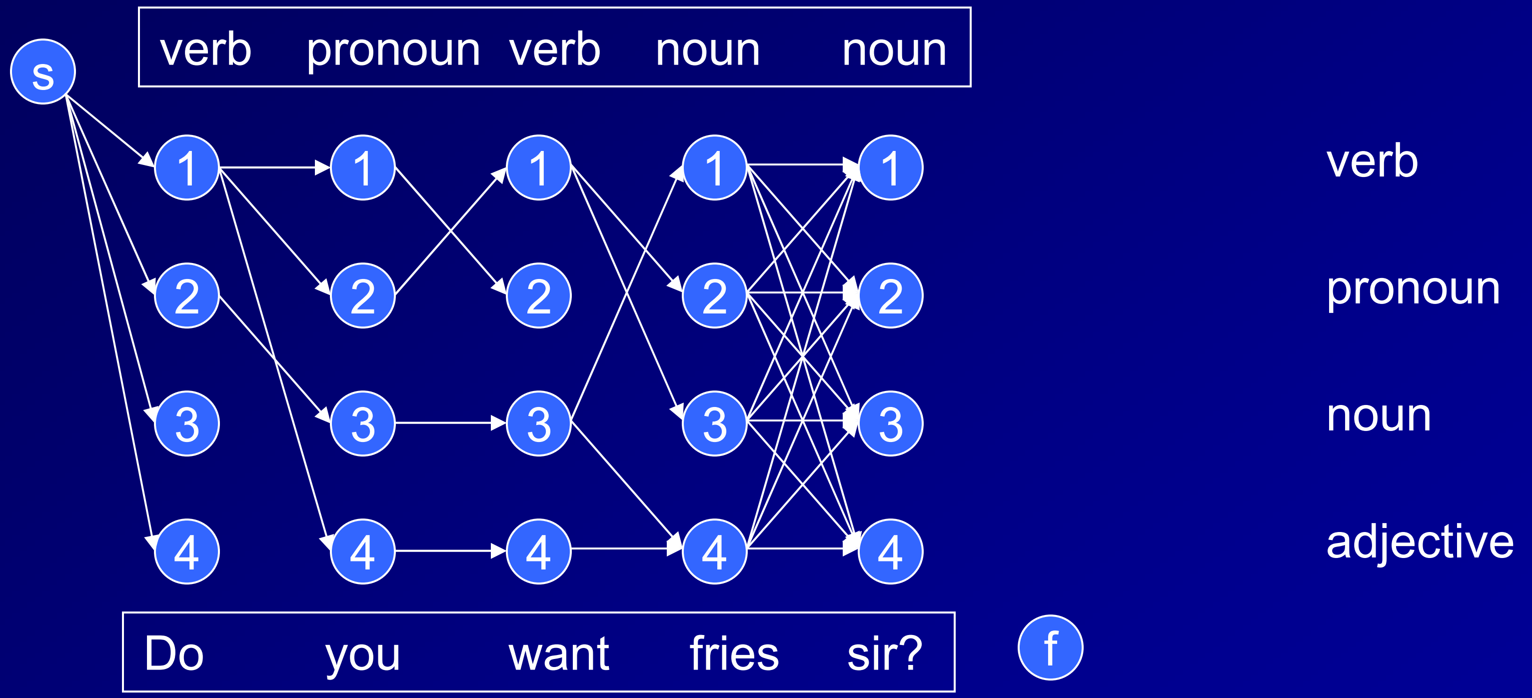
Compute successors

Finding Most Likely Label Sequence: The Viterbi Algorithm



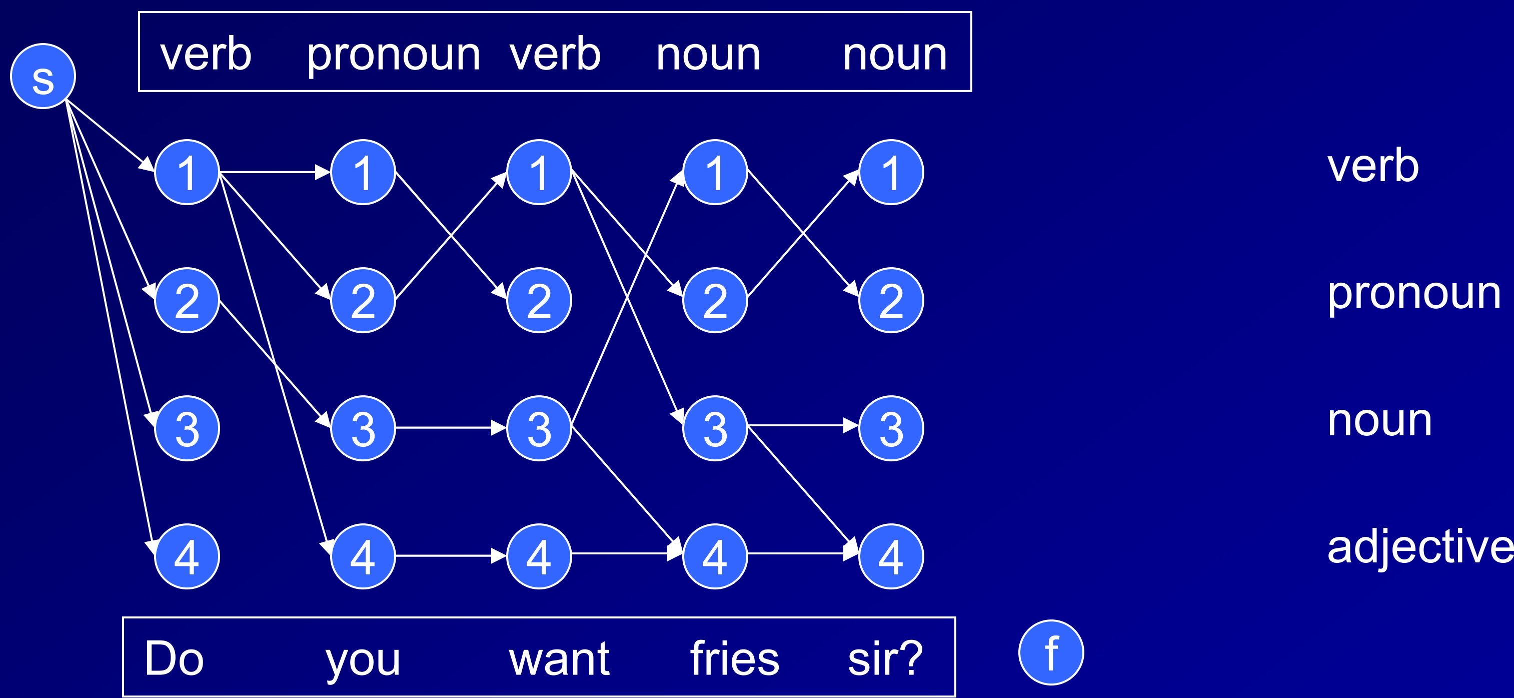
Compute and store shortest incoming arc at each node

Finding Most Likely Label Sequence: The Viterbi Algorithm



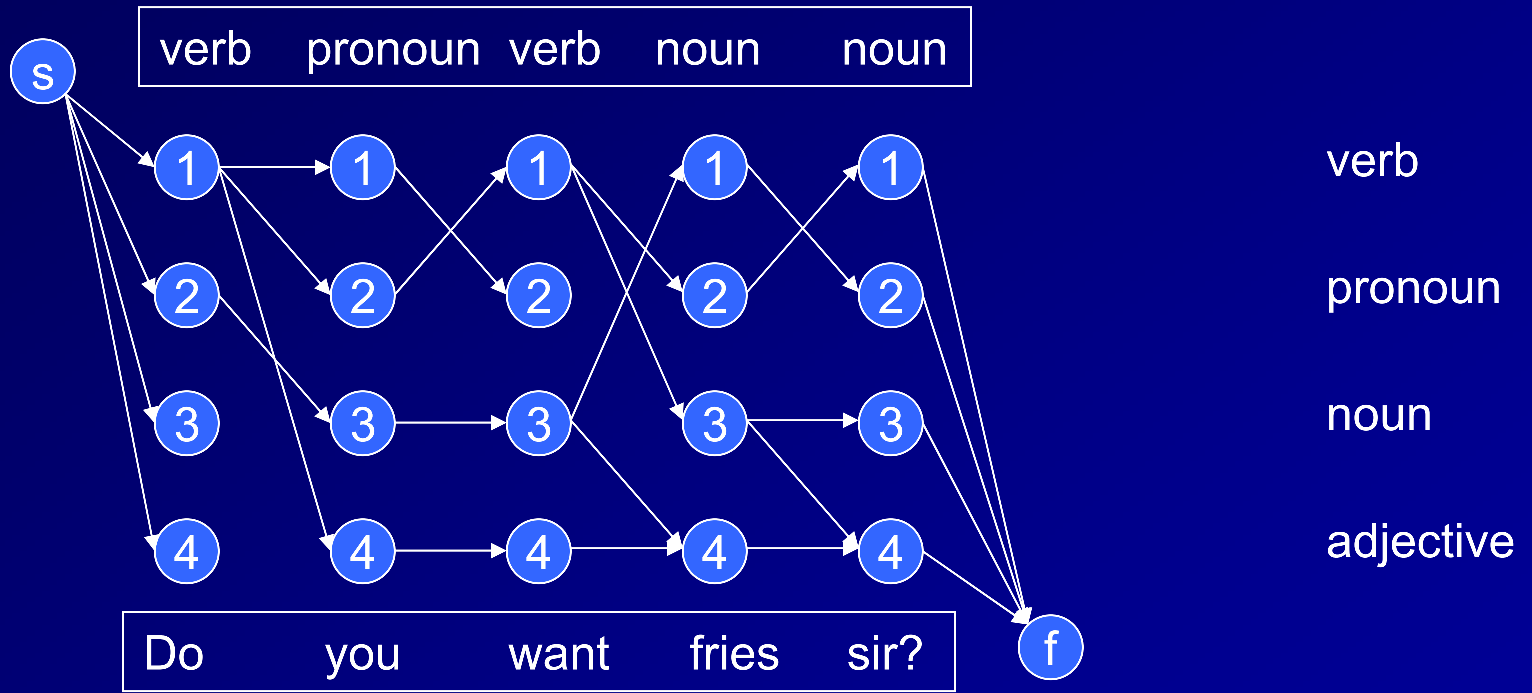
Compute successors…

Finding Most Likely Label Sequence: The Viterbi Algorithm



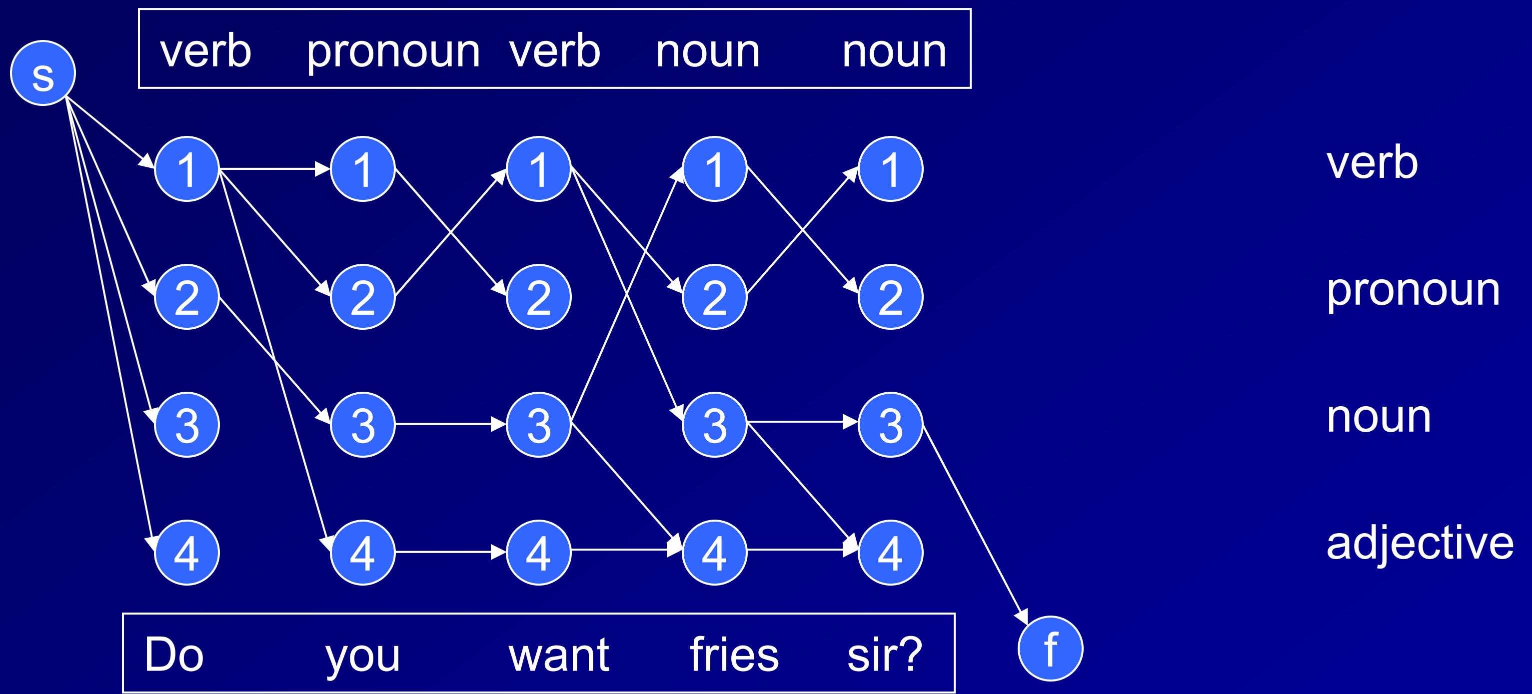
Compute and store shortest incoming edges

Finding Most Likely Label Sequence: The Viterbi Algorithm



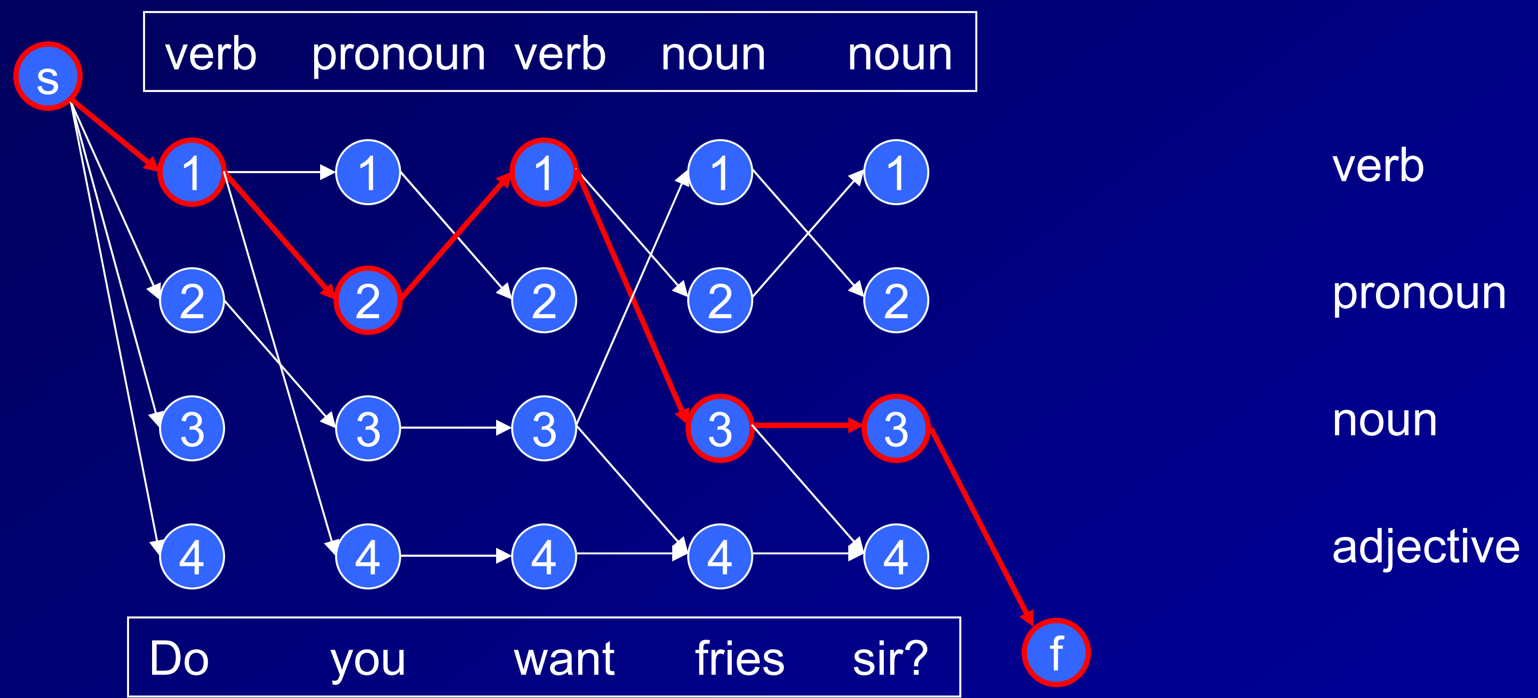
Compute successors (trivial)

Finding Most Likely Label Sequence: The Viterbi Algorithm



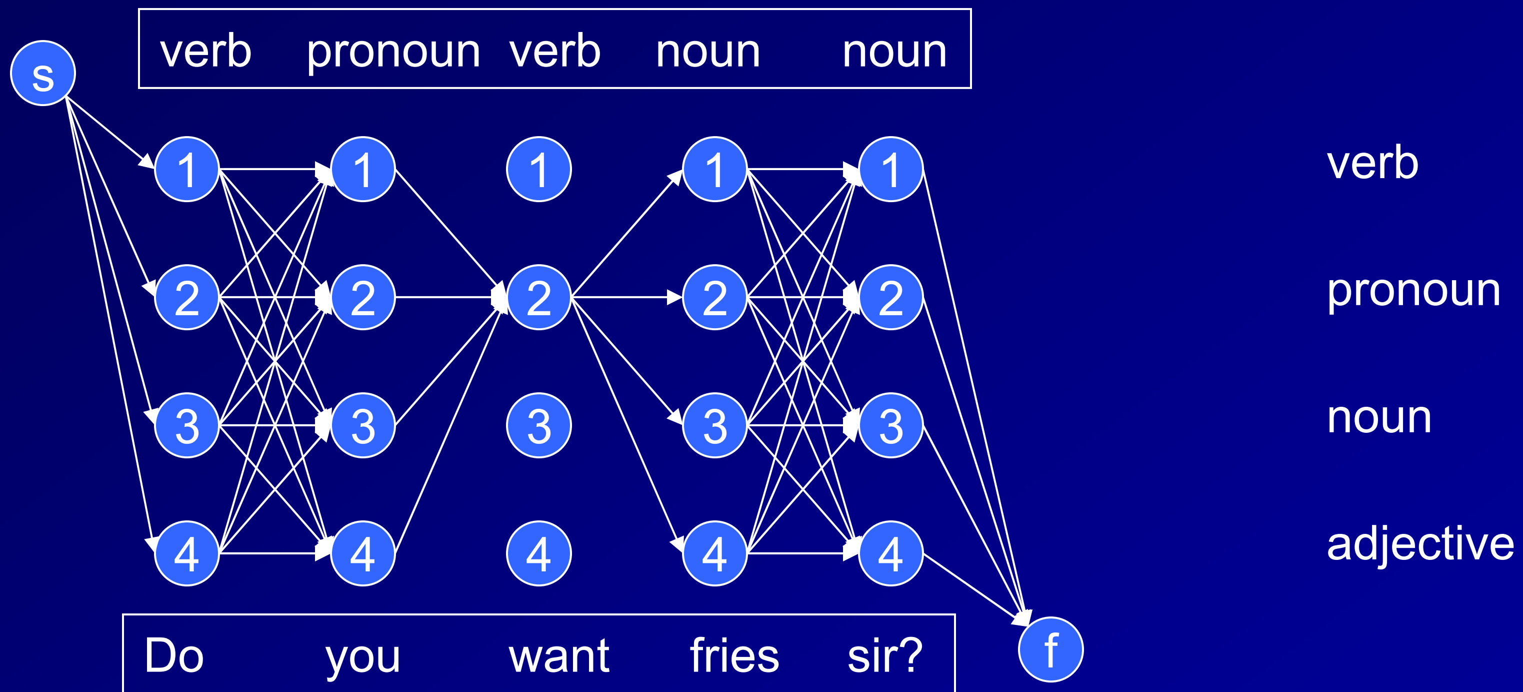
Compute best edge into f

Finding Most Likely Label Sequence: The Viterbi Algorithm



Now trace back along best incoming edges to recover the predicted Y sequence: “verb pronoun verb noun noun”

Finding the Most Likely Label at time t: $P(y_t \mid X)$



$P(y_3=2 \mid X)$ = probability of reaching $y_3=2$ from the start * probability of getting from $y_3=2$ to the finish

Finding the most likely class at each time t

goal: compute $P(y_t \mid \mathbf{x}_1, \ldots, \mathbf{x}_T)$

$\propto \sum_{y_{1:t-1}} \sum_{y_{t+1:T}} P(y_1) \cdot P(\mathbf{x}_1|y_1) \cdot P(y_2|y_1) \cdot P(\mathbf{x}_2|y_2) \cdots P(y_T \mid y_{T-1}) \cdot P(\mathbf{x}_T|y_T)$

$\propto \sum_{y_{1:t-1}} P(y_1) \cdot P(\mathbf{x}_1|y_1) \cdot P(y_2|y_1) \cdot P(\mathbf{x}_2|y_2) \cdots P(y_t|y_{t-1}) \cdot P(\mathbf{x}_t \mid y_t) \cdot$
$\sum_{y_{t+1:T}} P(y_{t+1}|y_t)\, P(\mathbf{x}_{t+1}|y_{t+1}) \cdots P(y_T \mid y_{T-1}) \cdot P(\mathbf{x}_T \mid y_T)$

$\propto \sum_{y_{t-1}} [\cdots \sum_{y_2} [\sum_{y_1} P(y_1) \cdot P(\mathbf{x}_1|y_1) \cdot P(y_2|y_1)] \cdot P(\mathbf{x}_2|y_2) \cdot \quad P(y_3|y_2)] \cdots$
$P(y_t|y_{t-1})] \cdot$
$P(\mathbf{x}_t|y_t) \cdot$
$\sum_{y_{t+1}} [P(y_{t+1}|y_t) \cdot P(\mathbf{x}_{t+1}|y_{t+1}) \cdots \sum_{y_{T-1}} [P(y_{T-1}|y_{T-2}) \cdot P(\mathbf{x}_{T-1}|y_{T-1}) \cdot \sum [P(y_T|y_{T-1}) \cdot P(\mathbf{x}_T \mid y_T)]] \cdots]$

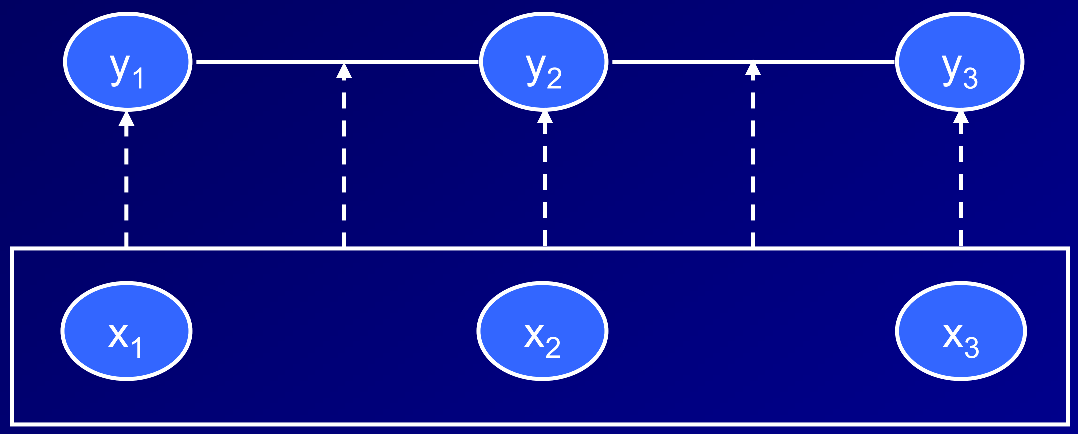
Forward-Backward Algorithm

- $\alpha_t(y_t) = \sum_{y_{t-1}} P(y_t \mid y_{t-1}) \cdot P(\mathbf{x}_t \mid y_t) \cdot \alpha_{t-1}(y_{t-1})$
  - This is the sum over the arcs coming into $y_t = k$
  - It is computed "forward" along the sequence and stored in the trellis
- $\beta_t(y_t) = \sum_{y_{t+1}} P(y_{t+1} | y_t) \cdot P(\mathbf{x}_{t+1} \mid y_{t+1}) \cdot \beta_{t+1}(y_{t+1})$
  - It is computed "backward" along the sequence and stored in the trellis
- $P(y_t \mid X) = \alpha_t(y_t)\ \beta_t(y_t) / [\sum_k \alpha_t(k)\ \beta_t(k)]$

Training Hidden Markov Models

- If the inputs and outputs are fully-observed, this is extremely easy:
- $P(y_1=k)$ = [# examples with $y_1=k$] / m
- $P(y_t=k \mid y_{t-1} = k') =$
  [# <k,k'> transitions] / [# of times $y_t = k$]
- $P(x_j = v \mid y = k) =$
  [# times $y=k$ and $x_j=v$] / [# times $y_t = k$]
- Should apply Laplace corrections to these estimates

Conditional Random Fields



- The y_t's form a Markov Random Field conditioned on X: P(Y|X)

Lafferty, McCallum, & Pereira (2001)

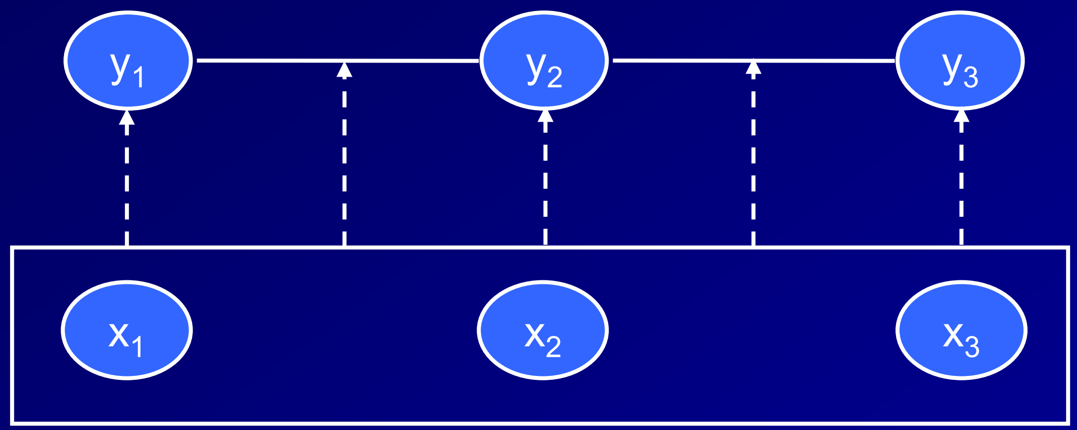
Markov Random Fields

- Graph $G = (V,E)$
  - Each vertex $v \in V$ represents a random variable y_v.
  - Each edge represents a direct probabilistic dependency.
- $P(Y) = 1/Z \exp [\sum_c \Psi_c(c(Y))]$
  - c indexes the cliques in the graph
  - Ψ_c is a potential function
  - $c(Y)$ selects the random variables participating in clique c.

A Simple MRF

- Cliques:
  - singletons: $\{y_1\}, \{y_2\}, \{y_3\}$
  - pairs (edges); $\{y_1,y_2\}, \{y_2,y_3\}$
- $P(\langle y_1,y_2,y_3\rangle) = 1/Z \exp[\Psi_1(y_1) + \Psi_2(y_2) +$
  $\Psi_3(y_3) + \Psi_{12}(y_1,y_2) + \Psi_{23}(y_2,y_3)]$

CRF Potential Functions are Conditioned on X



- $\Psi_t(y_t,X)$: how compatible is y_t with X?
- $\Psi_{t,t-1}(y_t,y_{t-1},X)$: how compatible is a transition from y_{t-1} to y_t with X?

CRF Potentials are Log Linear Models

- $\Psi_t(y_t, X) = \sum_b \beta_b \, g_b(y_t, X)$
- $\Psi_{t,t+1}(y_t, y_{t+1}, X) = \sum_a \lambda_a \, f_a(y_t, y_{t+1}, X)$

- where g_b and f_a are user-defined boolean functions ("features")
  - Example: g_{23} = [x_t = "o" and y_t = /@/]

- we will lump them together as

  $\Psi_t(y_t, y_{t+1}, X) = \sum_a \lambda_a \, f_a(y_t, y_{t+1}, X)$

Making Predictions with CRFs

- Viterbi and Forward-Backward algorithms can be applied exactly as for HMMs

Training CRFs

- Let $\theta = \{\beta_1, \beta_2, \ldots, \lambda_1, \lambda_2, \ldots\}$ be all of our parameters
- Let F_θ be our CRF, so $F_\theta(Y,X) = P(Y|X)$
- Define the loss function $L(Y,F_\theta(Y,X))$ to be the Negative Log Likelihood

  $L(Y,F_\theta(Y,X)) = -\log F_\theta(Y,X)$
- Goal: Find θ to minimize loss (maximize likelihood)
- Algorithm: Gradient Descent

Gradient Computation

$$
\begin{aligned}
g_q &= \frac{\partial}{\partial \lambda_q} \log P(Y|X) \\
&= \frac{\partial}{\partial \lambda_q} \log \frac{\prod_t \exp \Psi_t(y_t, y_{t-1}, X)}{Z} \\
&= \frac{\partial}{\partial \lambda_q} \sum_t \Psi_t(y_t, y_{t-1}, X) - \log Z \\
&= \sum_t \frac{\partial}{\partial \lambda_q} \sum_a \lambda_a f_a(y_t, y_{t-1}, X) - \frac{\partial}{\partial \lambda_q} \log Z \\
&= \sum_t f_q(y_t, y_{t-1}, X) - \frac{\partial}{\partial \lambda_q} \log Z
\end{aligned}
$$

Gradient of Z

$$
\begin{aligned}
\frac{\partial}{\partial \lambda_q} \log Z &= \frac{1}{Z}\frac{\partial Z}{\partial \lambda_q} \\
&= \frac{1}{Z}\frac{\partial}{\partial \lambda_q}\sum_{Y'}\prod_t \exp \Psi_t(y'_t, y'_{t-1}, X) \\
&= \frac{1}{Z}\frac{\partial}{\partial \lambda_q}\sum_{Y'}\exp \sum_t \Psi_t(y'_t, y'_{t-1}, X) \\
&= \frac{1}{Z}\sum_{Y'}\exp\left[\sum_t \Psi_t(y'_t, y'_{t-1}, X)\right]\sum_t \frac{\partial}{\partial \lambda_q}\Psi_t(y'_t, y'_{t-1}, X) \\
&= \sum_{Y'}\frac{\exp\left[\sum_t \Psi_t(y'_t, y'_{t-1}, X)\right]}{Z}\sum_t \frac{\partial}{\partial \lambda_q}\sum_a \lambda_a f_a(y'_t, y'_{t-1}, X) \\
&= \sum_{Y'} P(Y'|X)\left[\sum_t f_q(y'_t, y'_{t-1}, X)\right]
\end{aligned}
$$

Gradient Computation

$$g_q \;=\; \sum_t f_q(y_t, y_{t-1}, X) - \sum_{Y'} P(Y'|X)\left[\sum_t f_q(y'_t, y'_{t-1}, X)\right]$$

Number of times feature q is true minus the <u>expected</u> number of times feature q is true. This can be computed via the forward backward algorithm. First, apply forward-backward to compute $P(y_{t-1}, y_t \mid X)$.

$$P(y_{t-1}, y_t|X) \;=\; \frac{1}{Z}\sum_{y_t}\sum_{y_{t-1}} \alpha_{t-1}(y_{t-1}) \cdot \exp\Psi(y_t, y_{t-1}, X) \cdot \beta_t(y_t)$$

Then compute the gradient with respect to each λ_q

$$g_q \;=\; \sum_t f_q(y_t, y_{t-1}, X) - \sum_{y_t}\sum_{y_{t-1}} P(y_{t-1}, y_t|X) f_q(y_t, y_{t-1}, X)$$

Discriminative Methods

- Learn a discriminant function to which the Viterbi algorithm can be applied
  - "just get the right answer"
- Methods:
  - Averaged perceptron (Collins)
  - Hidden Markov SVMs (Altun, et al.)
  - Max Margin Markov Nets (Taskar, et al.)

Collins' Perceptron Method

- If we ignore the global normalizer in the CRF, the score for a label sequence Y given an input sequence X is

$$score(Y) \;=\; \sum_t \sum_a \lambda_a f_a(y_{t-1}, y_t, X)$$

- Collin's approach is to adjust the weights λ_a so that the correct label sequence gets the highest score according to the Viterbi algorithm

Sequence Perceptron Algorithm

- Initialize weights $\lambda_a = 0$
- For $\ell = 1, \ldots, L$ do
  - For each training example (X_i, Y_i)
    - apply Viterbi algorithm to find the path $\hat{Y}$ with the highest score
    - for all *a*, update λ_a according to

      $\lambda_a := \lambda_a + \sum_t [f_a(y_t, y_{t-1}, X) - f_a(\hat{y}_t, \hat{y}_{t-1}, X)]$
- Compares the "viterbi path" to the "correct path". Note that no update is made if the viterbi path is correct.

Averaged Perceptron

- Let $\lambda_a^{\ell,i}$ be the value of λ_a after processing training example *i* in iteration ℓ
- Define λ_a^* = the average value of λ_a = $1/(LN) \sum_{\ell,i} \lambda_a^{\ell,i}$
- Use these averaged weights in the final classifier

Collins Part-of-Speech Tagging with Averaged Sequence Perceptron

- Without averaging: 3.68% error
  - 20 iterations
- With averaging: 2.93% error
  - 10 iterations

Hidden Markov SVM

- Define a kernel between two input values **x** and **x**': k(**x**,**x**').
- Define a kernel between (X,Y) and (X',Y') as follows:

K((X,Y), (X',Y')) =

$$\sum_{s,t} I[y_{s-1} = y'_{t-1} \ \& \ y_s = y'_t] + I[y_s = y'_t] \, k(\mathbf{x}_s, \mathbf{x'}_t)$$

Number of (y_{t-1}, y_t) transitions that they share + Number of matching labels (weighted by similarity between the **x** values)

Dual Form of Linear Classifier

- Score(Y|X) =

  $\sum_j \sum_a \alpha_j(Y_a)\, K((X_j, Y_a), (X, Y))$

  a indexes "support vector" label sequences Y_a
- Learning algorithm finds
  - set of Y_a label sequences
  - weight values $\alpha_j(Y_a)$

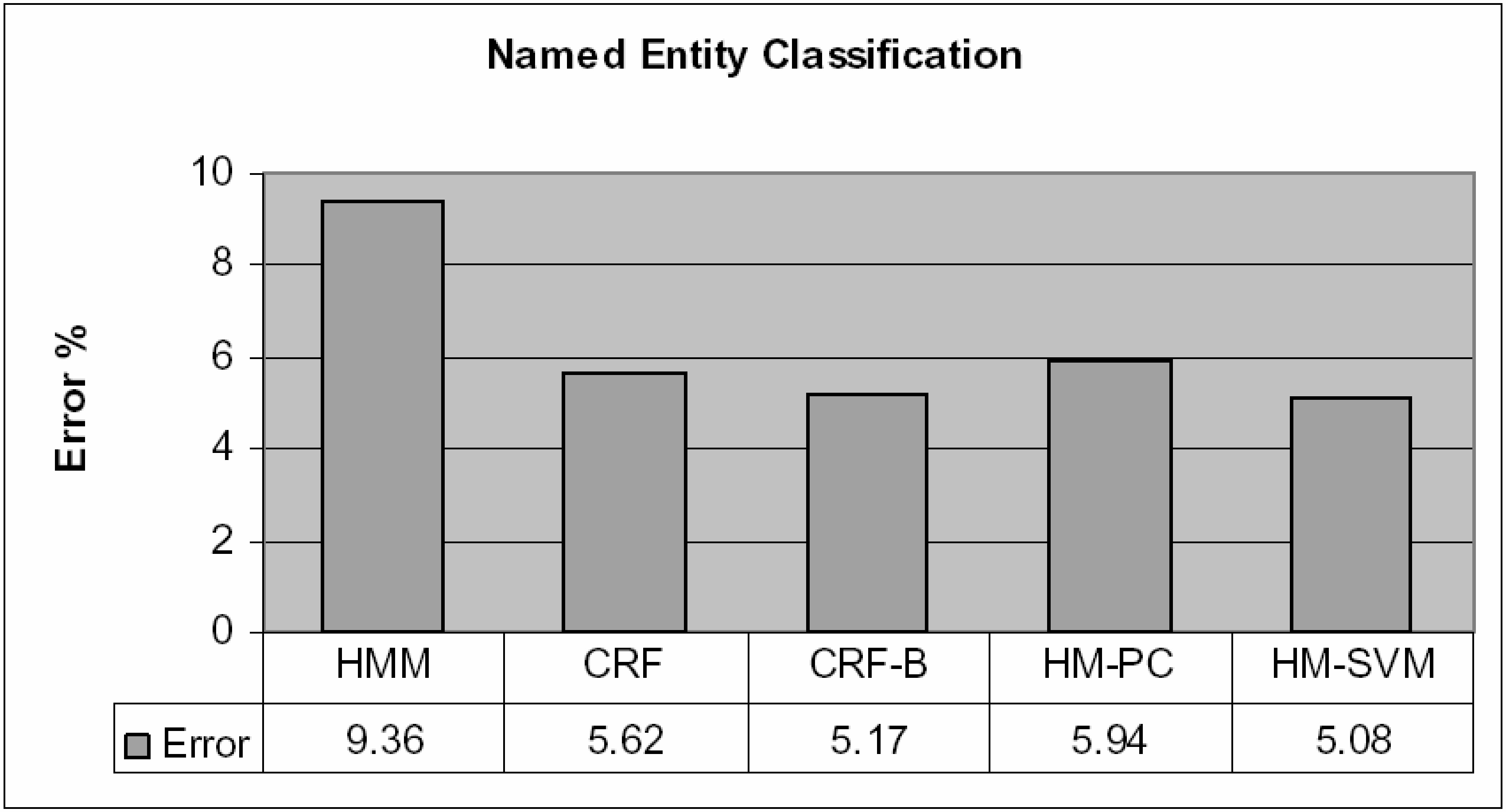
Dual Perceptron Algorithm

- Initialize $\alpha_j = 0$
- For ℓ from 1 to L do
  - For i from 1 to N do
    - $\hat{Y} = \text{argmax}_Y \text{ Score}(Y \mid X_i)$
    - if $\hat{Y} \neq Y_i$ then
      - $\alpha_i(Y_i) = \alpha_i(Y_i) + 1$
      - $\alpha_i(\hat{Y}) = \alpha_i(\hat{Y}) - 1$

Hidden Markov SVM Algorithm

- For all i initialize
  - $S_i = \{Y_i\}$ set of "support vector sequences" for i
  - $\alpha_i(Y)=0$ for all Y in S_i
- For ℓ from 1 to L do
  - For i from 1 to N do
    - $\hat{Y} = \text{argmax}_{Y \neq Y_i}$ Score(Y | X_i)
    - If Score(Y_i | X_i) < Score($\hat{Y}$ | X_i)
      - Add $\hat{Y}$ to S_i
      - Solve quadratic program to optimize the $\alpha_i(Y)$ for all Y in S_i to maximize the margin between Y_i and all of the other Y's in S_i
      - If $\alpha_i(Y) = 0$, delete Y from S_i

Altun et al. comparison



Maximum Margin Markov Networks

- Define SVM-like optimization problem to maximize the <u>per time step</u> margin
- Define

  $\Delta F(X_i, Y_i, \hat{Y}) = F(X_i, Y_i) - F(X_i, \hat{Y})$

  $\Delta Y(Y_i, \hat{Y}) = \sum_i I[\hat{y}_t \neq y_{it}]$
- MMM SVM formulation:

  $\min ||w||^2 + C \sum_i \xi_i$

  subject to

  $\mathbf{w} \cdot \Delta F(X_i, Y_i, \hat{Y}) \geq \Delta Y(Y_i, \hat{Y}) + \xi_i$ forall Y, forall i

Dual Form

maximize $\sum_i \sum_Y \alpha_i(\hat{Y})\, \Delta(Y_i, \hat{Y})$ –

$\frac{1}{2} \sum_i \sum_{\hat{Y}} \sum_j \sum_{\hat{Y}'} \alpha_i(\hat{Y})\, \alpha_j(\hat{Y}')\, [\Delta F(X_i,Y_i,\hat{Y}) \cdot \Delta F(X_j,Y_j,\hat{Y}')]$

subject to

$\sum_{\hat{Y}} \alpha_i(\hat{Y}) = C$ forall i

$\alpha_i(\hat{Y}) \geq 0$ forall i, forall $\hat{Y}$

Note that there are exponentially-many $\hat{Y}$ label sequences

Converting to a Polynomial-Sized Formulation

- Note the constraints:

  $\sum_{\hat{Y}} \alpha_i(\hat{Y}) = C$ forall i

  $\alpha_i(\hat{Y}) \geq 0$ forall i, forall $\hat{Y}$
- These imply that for each i, the $\alpha_i(\hat{Y})$ values are proportional to a probability distribution:

  $Q(\hat{Y} \mid X_i) = \alpha_i(\hat{Y}) / C$
- Because the MRF is a simple chain, this distribution can be factored into local distributions:

  $Q(\hat{Y} \mid X_i) = \prod_t Q(\hat{y}_{t-1}, \hat{y}_t \mid X_i)$
- Let $\mu_i(\hat{y}_{t-1}, \hat{y}_t)$ be the unnormalized version of Q

Reformulated Dual Form

$$\max \sum_i \sum_t \sum_{\widehat{y}_t} \mu_i(\widehat{y}_t) I[\widehat{y}_t \neq y_{i,t}] -$$
$$\frac{1}{2} \sum_{i,j} \sum_t \sum_{\widehat{y}_t, \widehat{y}_{t-1}} \sum_s \sum_{\widehat{y}'_s, \widehat{y}'_{s-1}} \mu_i(\widehat{y}_{t-1}, \widehat{y}_t) \mu_j(\widehat{y}'_{s-1}, \widehat{y}'_s)$$
$$\Delta F(\widehat{y}_{t-1}, \widehat{y}_t, X_i) \cdot \Delta F(\widehat{y}'_{s-1}, \widehat{y}'_s, X_j)$$

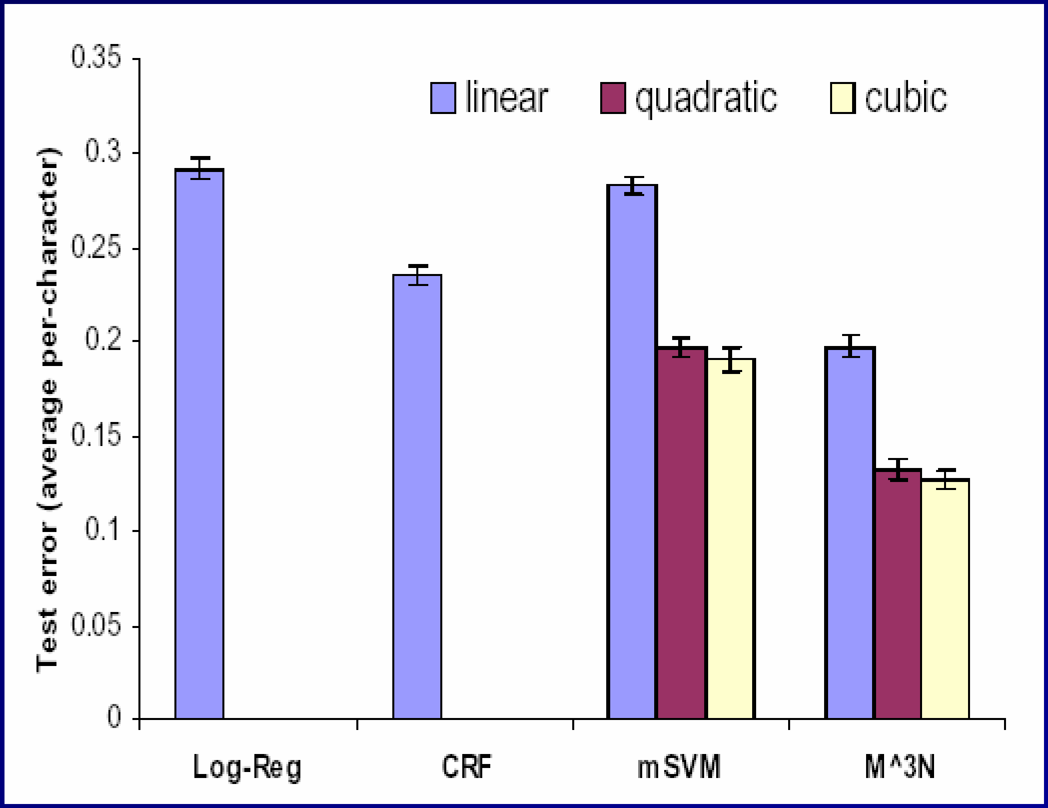
subject to

$$\begin{aligned} \sum_{\widehat{y}_{t-1}} \mu_i(\widehat{y}_{t-1}, \widehat{y}_t) &= \mu_i(\widehat{y}_t) \\ \sum_{\widehat{y}_t} \mu_i(\widehat{y}_t) &= C \\ \mu_i(\widehat{y}_{t-1}, \widehat{y}_t) &\geq 0 \end{aligned}$$

Variables in the Dual Form

- $\mu_i(k,k')$ for each training example i and each possible class labels k, k': $O(NK^2)$
- $\mu_i(k)$ for each trianing example i and possible class label k: $O(NK)$
- Polynomial!

Taskar et al. comparison Handwriting Recognition



log-reg: logistic regression sliding window

CRF:

mSVM: multiclass SVM sliding window

M^3N: max margin markov net

Current State of the Art

- Discriminative Methods give best results
  - not clear whether they scale
  - published results all involve small numbers of training examples and very long training times
- Work is continuing on making CRFs fast and practical
  - new methods for training CRFs
  - potentially extendable to discriminative methods