Language and Statistics II

Lecture 17: Discriminative Training, part III

Noah Smith

Lecture Overview

- Formal problem from assignment 3
- ~~MIRA~~
- Kernel for trees and sequences (Collins)
- Discriminative reranking
- Transformation-based learning (if time)

HMM as a PCFG

HMM $H = \langle \Sigma, Q, q_0, F, e:(Q \setminus F) \times \Sigma \rightarrow \mathbb{P}, t:Q \times Q \rightarrow \mathbb{P} \rangle$

PCFG $G = \langle \Sigma, N, n_0, r:N \times (N \cup \Sigma)^* \rightarrow \mathbb{P} \rangle$

Let:

$\Sigma = \Sigma$

$N = Q$

$n_0 = q_0$

$r(q, \langle s, q' \rangle) = e(q, s) \cdot t(q, q') \quad \text{if } q' \notin F$

$r(q, \langle s \rangle) = e(q, s) \cdot \bigoplus_{q':q' \in F} t(q, q')$

A Parsing Algorithm for PCFGs (with rank ≤ 2)

$C(s_i, i, i) = 1$

$C(X, i, k) = \max(\max_Y C(Y, I, k) \times r(X, \langle Y \rangle),$

$\max_{j, Y, Z} C(Y, i, j) \times C(Z, j+1, k) \times r(X, \langle Y, Z \rangle))$

$\text{goal} = C(N_0, 1, |\mathbf{s}|)$

$\text{priority}(C(X, i, j)) = |\mathbf{s}| - (j - i)$

$r(q, \langle s, q' \rangle) = e(q, s) \cdot t(q, q')$ if $q' \notin F$

$r(q, \langle s \rangle) = e(q, s) \cdot \max_{q' : q' \in F} t(q, q')$

A Parsing Algorithm for PCFGs (with rank ≤ 2)

$C(s_i, i, i) = 1$

$C(X, i, k) = \max(\max_Y C(Y, i, k) \times r(X, \langle Y \rangle),$

$\max_{j, Y, Z} C(Y, i, j) \times C(Z, j+1, k) \times r(X, \langle Y, Z \rangle))$

$\text{goal} = C(N_0, 1, |\mathbf{s}|)$

$\text{priority}(C(X, i, j)) = |\mathbf{s}| - (j - i)$

Notice: When binary rule fires, Y is always in Σ.

$r(q, \langle s, q' \rangle) = e(q, s) \cdot t(q, q')$ if $q' \notin F$

$r(q, \langle s \rangle) = e(q, s) \cdot \max_{q' : q' \in F} t(q, q')$

A Parsing Algorithm for PCFGs (with rank ≤ 2)

$C(s_i, i, i) = 1$

$C(X, i, k) = \max(\max_Y C(Y, i, k) \times r(X, \langle Y \rangle),$

$\max_{Y, Z} C(Y, i, i) \times C(Z, i+1, k) \times r(X, \langle Y, Z \rangle))$

$\text{goal} = C(N_0, 1, |\mathbf{s}|)$

$\text{priority}(C(X, i, j)) = |\mathbf{s}| - (j - i)$

Notice: When binary rule fires, Y is always in Σ.

So j = i.

$r(q, \langle s, q' \rangle) = e(q, s) \cdot t(q, q')$ if $q' \notin F$

$r(q, \langle s \rangle) = e(q, s) \cdot \max_{q' : q' \in F} t(q, q')$

A Parsing Algorithm for PCFGs (with rank ≤ 2)

$C(s_i, i, i) = 1$

$C(X, i, k) = \max(\max_Y C(Y, i, k) \times r(X, \langle Y \rangle),$

$\max_{Y, Z} C(Y, i, i) \times C(Z, i+1, k) \times r(X, \langle Y, Z \rangle))$

$\text{goal} = C(N_0, 1, |\mathbf{s}|)$

$\text{priority}(C(X, i, j)) = |\mathbf{s}| - (j - i)$

$r(q, \langle s, q' \rangle) = e(q, s) \cdot t(q, q')$ if $q' \notin F$

$r(q, \langle s \rangle) = e(q, s) \cdot \max_{q' : q' \in F} t(q, q')$

A Parsing Algorithm for PCFGs (with rank ≤ 2)

$C(s_i, i, i) = 1$

$C(X, i, k) = \max(\max_Y C(Y, i, k) \times r(X, \langle Y \rangle),$

$\max_{Y, Z} C(Y, i, i) \times C(Z, i+1, k) \times r(X, \langle Y, Z \rangle))$

$\text{goal} = C(N_0, 1, |\mathbf{s}|)$

$\text{priority}(C(X, i, j)) = |\mathbf{s}| - (j - i)$

Notice: When binary rule fires, Y always = s_i.

$r(q, \langle s, q' \rangle) = e(q, s) \cdot t(q, q')$ if $q' \notin F$

$r(q, \langle s \rangle) = e(q, s) \cdot \max_{q' : q' \in F} t(q, q')$

A Parsing Algorithm for PCFGs (with rank ≤ 2)

$C(s_i, i, i) = 1$

$C(X, i, k) = \max(\max_Y C(Y, i, k) \times r(X, \langle Y \rangle),$

$\max_Z C(s_i, i, i) \times C(Z, i+1, k) \times r(X, \langle s_i, Z \rangle))$

$goal = C(N_0, 1, |\mathbf{s}|)$

$priority(C(X, i, j)) = |\mathbf{s}| - (j - i)$

Notice: When binary rule fires, Y always = s_i.

So substitute.

$r(q, \langle s, q' \rangle) = e(q, s) \cdot t(q, q')$ if $q' \notin F$

$r(q, \langle s \rangle) = e(q, s) \cdot \max_{q' : q' \in F} t(q, q')$

A Parsing Algorithm for PCFGs (with rank ≤ 2)

$C(s_i, i, i) = 1$

$C(X, i, k) = \max(\max_Y C(Y, i, k) \times r(X, \langle Y \rangle),$

$\max_Z \sout{C(s_i, i, i) \times} C(Z, i+1, k) \times r(X, \langle s_i, Z \rangle))$

$goal = C(N_0, 1, |\mathbf{s}|)$

$priority(C(X, i, j)) = |\mathbf{s}| - (j - i)$

Notice: When binary rule fires, Y always = s_i.

And $C(s_i, i, i) = 1$.

$r(q, \langle s, q' \rangle) = e(q, s) \cdot t(q, q')$ if $q' \notin F$

$r(q, \langle s \rangle) = e(q, s) \cdot \max_{q' : q' \in F} t(q, q')$

A Parsing Algorithm for PCFGs (with rank ≤ 2)

$C(s_i, i, i) = 1$

$C(X, i, k) = \max(\max_Y C(Y, i, k) \times r(X, \langle Y \rangle),$

$\max_Z C(Z, i+1, k) \times r(X, \langle s_i, Z \rangle))$

$\text{goal} = C(N_0, 1, |\mathbf{s}|)$

$\text{priority}(C(X, i, j)) = |\mathbf{s}| - (j - i)$

$r(q, \langle s, q' \rangle) = e(q, s) \cdot t(q, q')$ if $q' \notin F$

$r(q, \langle s \rangle) = e(q, s) \cdot \max_{q' : q' \in F} t(q, q')$

A Parsing Algorithm for PCFGs (with rank ≤ 2)

$C(s_i, i, i) = 1$

$C(X, i, k) = \max(\max_Y C(Y, i, k) \times r(X, \langle Y \rangle),$

$\max_Z C(Z, i+1, k) \times r(X, \langle s_i, Z \rangle))$

$goal = C(N_0, 1, |\mathbf{s}|)$

$priority(C(X, i, j)) = |\mathbf{s}| - (j - i)$

Notice: When unary rule fires, Y is always in Σ and = s_i.

$r(q, \langle s, q' \rangle) = e(q, s) \cdot t(q, q')$ if $q' \notin F$

$r(q, \langle s \rangle) = e(q, s) \cdot \max_{q' : q' \in F} t(q, q')$

A Parsing Algorithm for PCFGs (with rank ≤ 2)

$C(s_i, i, i) = 1$

$C(X, i, k) = \max(C(s_i, i, k) \times r(X, \langle s_i \rangle),$

$\max_Z C(Z, i+1, k) \times r(X, \langle s_i, Z \rangle))$

$\text{goal} = C(N_0, 1, |\mathbf{s}|)$

$\text{priority}(C(X, i, j)) = |\mathbf{s}| - (j - i)$

Notice: When unary rule fires, Y is always in Σ and = s_i.

So substitute.

$r(q, \langle s, q' \rangle) = e(q, s) \cdot t(q, q')$ if $q' \notin F$

$r(q, \langle s \rangle) = e(q, s) \cdot \max_{q' : q' \in F} t(q, q')$

A Parsing Algorithm for PCFGs (with rank ≤ 2)

$C(s_i, i, i) = 1$

$C(X, i, k) = \max(C(s_i, i, k) \times r(X, \langle s_i \rangle),$
$\quad \max_Z C(Z, i+1, k) \times r(X, \langle s_i, Z \rangle))$

$\text{goal} = C(N_0, 1, |\mathbf{s}|)$

$\text{priority}(C(X, i, j)) = |\mathbf{s}| - (j - i)$

$r(q, \langle s, q' \rangle) = e(q, s) \cdot t(q, q')$ if $q' \notin F$

$r(q, \langle s \rangle) = e(q, s) \cdot \max_{q' : q' \in F} t(q, q')$

A Parsing Algorithm for PCFGs (with rank ≤ 2)

$C(s_i, i, i) = 1$

$C(X, i, k) = \max(C(s_i, i, k) \times r(X, \langle s_i \rangle),$

$\qquad \max_Z C(Z, i+1, k) \times r(X, \langle s_i, Z \rangle))$

$\text{goal} = C(N_0, 1, |\mathbf{s}|)$

$\text{priority}(C(X, i, j)) = |\mathbf{s}| - (j - i)$

Notice: k either = i (in the unary case) or |**s**| (in the binary case).

$r(q, \langle s, q' \rangle) = e(q, s) \cdot t(q, q')$ if $q' \notin F$

$r(q, \langle s \rangle) = e(q, s) \cdot \max_{q' : q' \in F} t(q, q')$

A Parsing Algorithm for PCFGs (with rank ≤ 2)

$C(s_i, i, i) = 1$

$C(X, i, i) = C(s_i, i, i) \times r(X, \langle s_i \rangle)$

$C(X, i, |\mathbf{s}|) = \max_Z C(Z, i+1, |\mathbf{s}|) \times r(X, \langle s_i, Z \rangle)$

$\text{goal} = C(N_0, 1, |\mathbf{s}|)$

$\text{priority}(C(X, i, j)) = |\mathbf{s}| - (j - i)$

Notice: k either = i (in the unary case) or |**s**| (in the binary case).

$r(q, \langle s, q' \rangle) = e(q, s) \cdot t(q, q')$ if $q' \notin F$

$r(q, \langle s \rangle) = e(q, s) \cdot \max_{q' : q' \in F} t(q, q')$

A Parsing Algorithm for PCFGs (with rank ≤ 2)

$C(s_i, i, i) = 1$

$C(X, i, i) = C(s_i, i, i) \times r(X, \langle s_i \rangle)$

$C(X, i, |\mathbf{s}|) = \max_Z C(Z, i+1, |\mathbf{s}|) \times r(X, \langle s_i, Z \rangle)$

$\text{goal} = C(N_0, 1, |\mathbf{s}|)$

$\text{priority}(C(X, i, j)) = |\mathbf{s}| - (j - i)$

$r(q, \langle s, q' \rangle) = e(q, s) \cdot t(q, q')$ if $q' \notin F$

$r(q, \langle s \rangle) = e(q, s) \cdot \max_{q' : q' \in F} t(q, q')$

A Parsing Algorithm for PCFGs (with rank ≤ 2)

$C(s_i, i, i) = 1$

$C(X, i, i) = C(s_i, i, i) \times r(X, \langle s_i \rangle)$

$C(X, i, |\mathbf{s}|) = \max_Z C(Z, i+1, |\mathbf{s}|) \times r(X, \langle s_i, Z \rangle)$

$\text{goal} = C(N_0, 1, |\mathbf{s}|)$

$\text{priority}(C(X, i, j)) = |\mathbf{s}| - (j - i)$

Notice: $C(s_i, i, i)$ is unnecessary in general; only used when $i = |\mathbf{s}|$.

$r(q, \langle s, q' \rangle) = e(q, s) \cdot t(q, q')$ if $q' \notin F$

$r(q, \langle s \rangle) = e(q, s) \cdot \max_{q' : q' \in F} t(q, q')$

A Parsing Algorithm for PCFGs (with rank ≤ 2)

$C(X, |\mathbf{s}|, |\mathbf{s}|) = r(X, \langle s_{|\mathbf{s}|} \rangle)$

$C(X, i, |\mathbf{s}|) = \max_Z C(Z, i+1, |\mathbf{s}|) \times r(X, \langle s_i, Z \rangle)$

$\text{goal} = C(N_0, 1, |\mathbf{s}|)$

$\text{priority}(C(X, i, j)) = |\mathbf{s}| - (j - i)$

Notice: $C(s_i, i, i)$ is unnecessary in general; only used when $i = |\mathbf{s}|$.

$r(q, \langle s, q' \rangle) = e(q, s) \cdot t(q, q')$ if $q' \notin F$

$r(q, \langle s \rangle) = e(q, s) \cdot \max_{q' : q' \in F} t(q, q')$

A Parsing Algorithm for PCFGs (with rank ≤ 2)

$C(X, |\mathbf{s}|, |\mathbf{s}|) = r(X, \langle s_{|\mathbf{s}|} \rangle)$

$C(X, i, |\mathbf{s}|) = \max_Z C(Z, i+1, |\mathbf{s}|) \times r(X, \langle s_i, Z \rangle)$

$\text{goal} = C(N_0, 1, |\mathbf{s}|)$

$\text{priority}(C(X, i, j)) = |\mathbf{s}| - (j - i)$

$r(q, \langle s, q' \rangle) = e(q, s) \cdot t(q, q')$ if $q' \notin F$

$r(q, \langle s \rangle) = e(q, s) \cdot \max_{q' : q' \in F} t(q, q')$

A Parsing Algorithm for PCFGs (with rank ≤ 2)

$C(X, |\mathbf{s}|, |\mathbf{s}|) = r(X, \langle s_{|\mathbf{s}|} \rangle)$

$C(X, i, |\mathbf{s}|) = \max_Z C(Z, i+1, |\mathbf{s}|) \times r(X, \langle s_i, Z \rangle)$

$\text{goal} = C(N_0, 1, |\mathbf{s}|)$

$\text{priority}(C(X, i, j)) = |\mathbf{s}| - (j - i)$

Notice: third term of C(…) is always |**s**|.

$r(q, \langle s, q' \rangle) = e(q, s) \cdot t(q, q')$ if $q' \notin F$

$r(q, \langle s \rangle) = e(q, s) \cdot \max_{q' : q' \in F} t(q, q')$

A Parsing Algorithm for PCFGs (with rank ≤ 2)

$C(X, |\mathbf{s}|) = r(X, \langle s_{|\mathbf{s}|} \rangle)$

$C(X, i) = \max_Z C(Z, i+1) \times r(X, \langle s_i, Z \rangle)$

$\text{goal} = C(N_0, 1)$

$\text{priority}(C(X, i)) = |\mathbf{s}| - (|\mathbf{s}| - i) = i$

Notice: third term of C(…) is always |**s**|.

$r(q, \langle s, q' \rangle) = e(q, s) \cdot t(q, q')$ if $q' \notin F$

$r(q, \langle s \rangle) = e(q, s) \cdot \max_{q' : q' \in F} t(q, q')$

A Parsing Algorithm for PCFGs (with rank ≤ 2)

$C(X, |\mathbf{s}|) = r(X, \langle s_{|\mathbf{s}|} \rangle)$

$C(X, i) = \max_Z C(Z, i+1) \times r(X, \langle s_i, Z \rangle)$

$\text{goal} = C(N_0, 1)$

$\text{priority}(C(X, i)) = i$

$r(q, \langle s, q' \rangle) = e(q, s) \cdot t(q, q')$ if $q' \notin F$

$r(q, \langle s \rangle) = e(q, s) \cdot \max_{q' : q' \in F} t(q, q')$

Equivalent to Back-Viterbi!

$C(X, |\mathbf{s}|) = r(X, \langle s_{|\mathbf{s}|} \rangle)$
$C(X, i) = \max_Z C(Z, i+1) \times r(X, \langle s_i, Z \rangle)$
goal = $C(N_0, 1)$
priority($C(X, i)$) = i

$C(X, |\mathbf{s}|) = e(q, s) \times \max_{Z:Z \in F} t(X, Z)$
$C(X, i) = \max_{Z \notin F} C(Z, i+1) \times e(X, s_i) \times t(X, Z)$
goal = $C(q_0, 1)$
priority($C(X, i)$) = i

$r(q, \langle s, q' \rangle) = e(q, s) \cdot t(q, q')$ if $q' \notin F$
$r(q, \langle s \rangle) = e(q, s) \cdot \max_{q':q' \in F} t(q, q')$

Kernels and Dual Weights

- Recall that a **kernel** is a way of implicitly expanding our **feature set** by replacing the dot-product with new function of two vectors.

$$K\left(\mathbf{f}(x,y),\mathbf{f}(x',y')\right) = \mathbf{f}'(x,y)\cdot\mathbf{f}'(x',y')$$

- Recall also that dual formulation of linear models uses “support vectors” (sparse in SVMs, but holds for most parameter estimation methods):

$$\mathbf{w} = \sum_i \sum_{y'\in\mathrm{GEN}(y_i)} \alpha_{i,y'}\left(\mathbf{f}(x_i,y')\right) \qquad K\left(\mathbf{f}(x,y),\mathbf{w}\right) = \sum_i \sum_{y'\in\mathrm{GEN}(y_i)} \alpha_{i,y'} K\left(\mathbf{f}(x,y),\mathbf{f}(x_i,y')\right)$$

Specialized Kernels

- The kernel function *K* is just a measure of similarity.
  - Higher = more similar.
- A function of two objects (any objects!) is a kernel if it equates to taking a dot product in *some* feature space.
- Let’s reason backwards …
  - What feature spaces would we like?

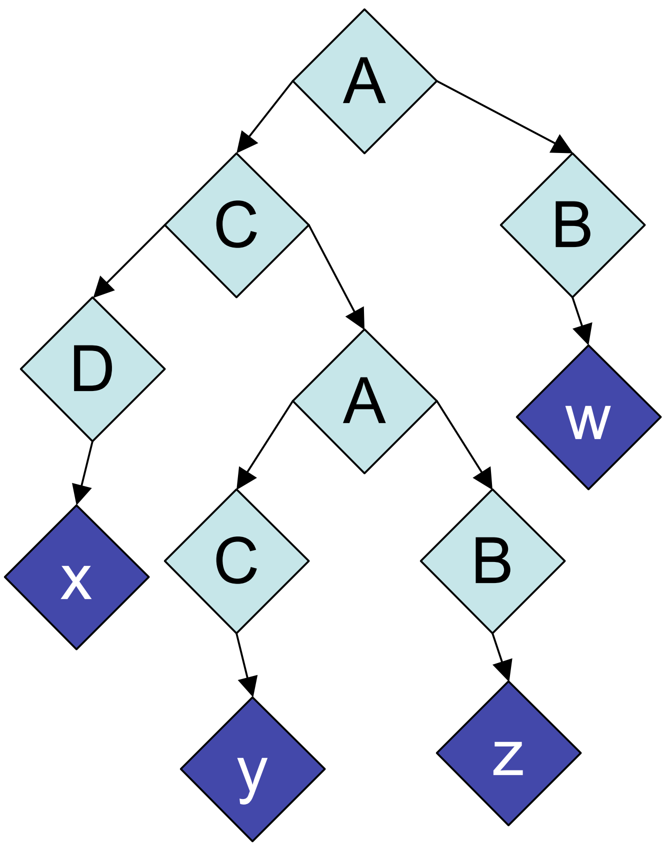
Tree Kernels (Collins and Duffy, 2002)

- The feature vector implied by the use of a CFG is a vector of **rule counts**.
- Implied kernel:

$$K(\tau_1, \tau_2) = \sum_{r \in G} \text{count}(r; \tau_1) \times \text{count}(r; \tau_2)$$

- Rules are just “bits” of structure.
- Scaling up: what if we could match up **all subtree** types?

Subtree Features



2 (A (C B))
1 (C (D A))
1 (D x)
1 (C y)
1 (B z)
1 (B w)
0 (D (B B)
0 (A (B C)
0 (B (C C)

1 (A ((C D A) B)
1 (A ((C D A) (B w))
1 (A ((C (D x) A) B)
1 (A ((C (D x) A) (B w))
1 (A ((C D (A C B)) (B w))
1 (A ((C (D x) (A C B)) (B w))
…
1 (A ((C (D x) (A (C y) (B z))) (B w))

“All Subtrees”

- Not a new idea.
  - Bod (1998 and before): “data-oriented parsing”
    - many tricks required - not efficient
  - Goodman (1996): convert DOP model to an approximating PCFG
- Collins and Duffy (2002): can implement this as a kernel!
  - Avoid the Really Big feature representation.
  - Train using discriminative methods.
- Note: subtrees contain full rules; can’t break just anywhere.

The All-Subtrees Kernel

$$K(\tau_1,\tau_2) = \sum_{\tau} \text{count}(\tau;\tau_1) \times \text{count}(\tau;\tau_2)$$

$$= \sum_{\tau} \left(\sum_{\tau_1^{\bullet} \subseteq \tau_1} \delta(\tau,\tau_1^{\bullet}) \right) \left(\sum_{\tau_2^{\bullet} \subseteq \tau_2} \delta(\tau,\tau_2^{\bullet}) \right)$$

$$= \sum_{\tau_1^{\bullet} \subseteq \tau_1} \sum_{\tau_2^{\bullet} \subseteq \tau_2} \sum_{\tau} \delta(\tau,\tau_1^{\bullet}) \delta(\tau,\tau_2^{\bullet})$$

$$= \sum_{n_1^i \subseteq \tau_1} \sum_{n_2^j \subseteq \tau_2} \underbrace{\left[\#\ \text{of matching subtrees rooted at } n_1^i, n_2^j \right]}_{\Delta\left(n_1^i, n_2^j\right)}$$

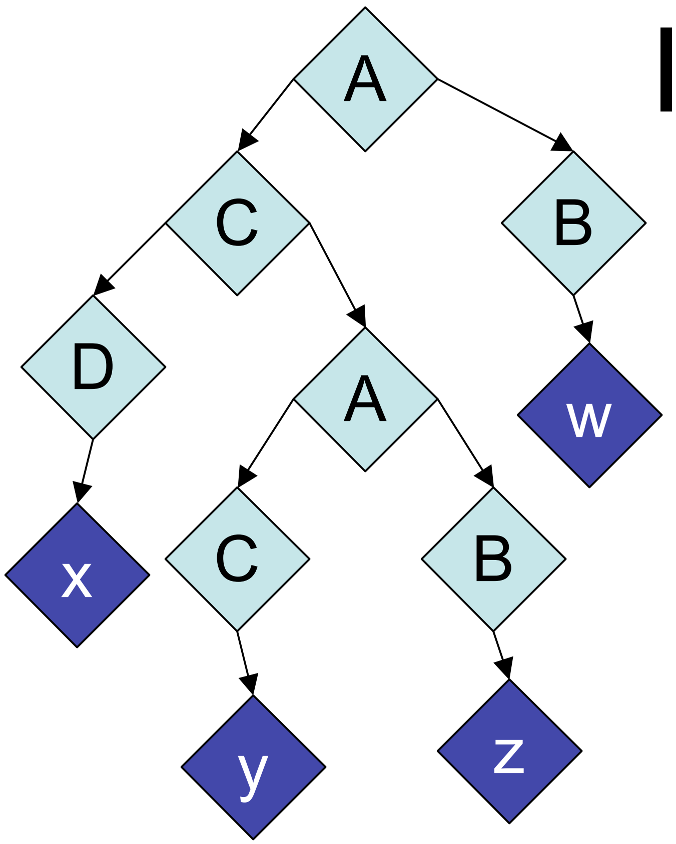
Summing over nodes in the two trees!

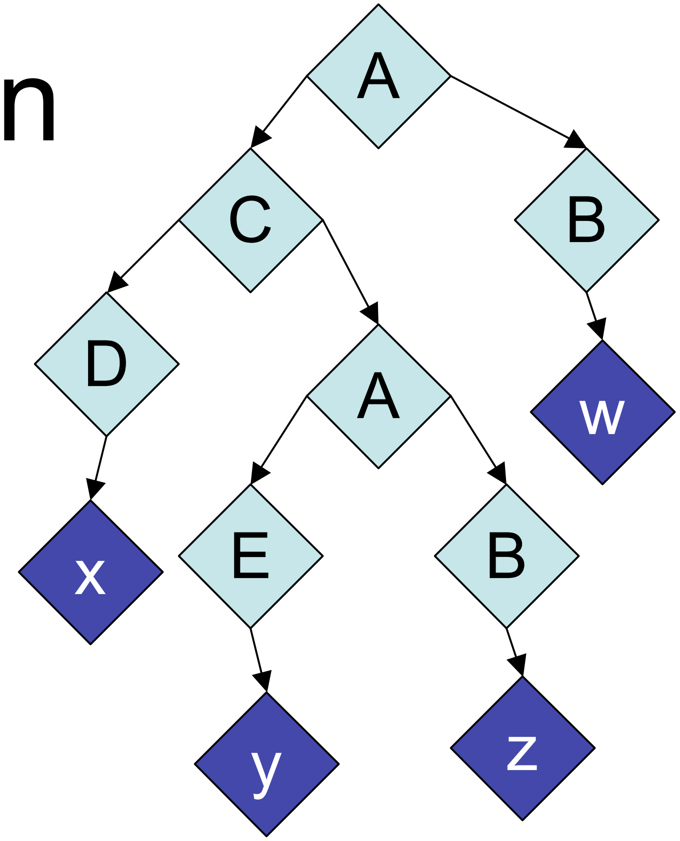
Dynamic Program

$$\Delta\left(n_1^i, n_2^j\right) = \begin{cases} 0 & \text{if different productions at roots} \\ 1 & \text{if same production and preterminal roots} \\ \prod_k \left(1 + \Delta\left(n_1^{\mathrm{kid}(i,k)}, n_2^{\mathrm{kid}(j,k)}\right)\right) & \text{if same production and not preterminal roots} \end{cases}$$

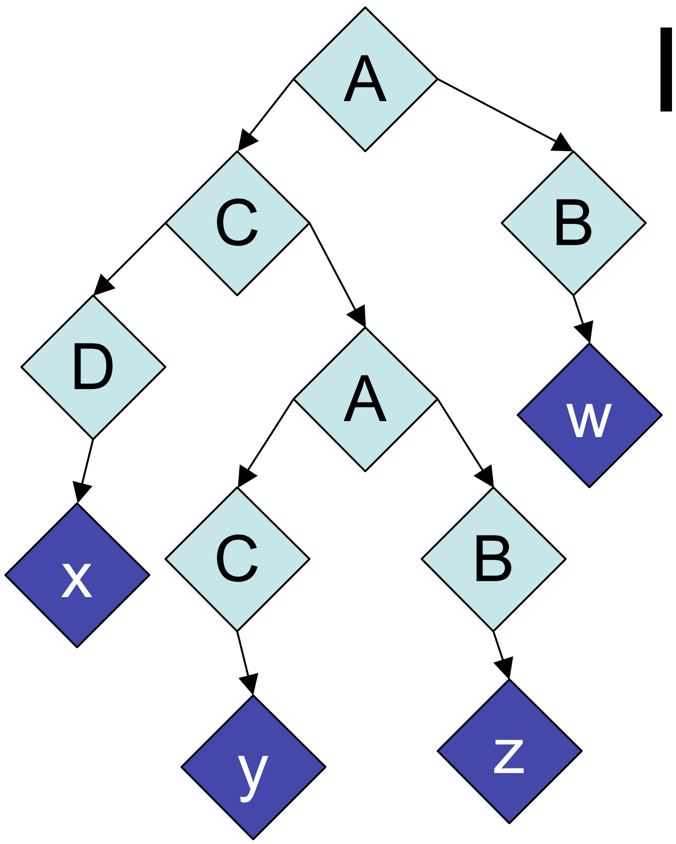
Thought question: what's the runtime?

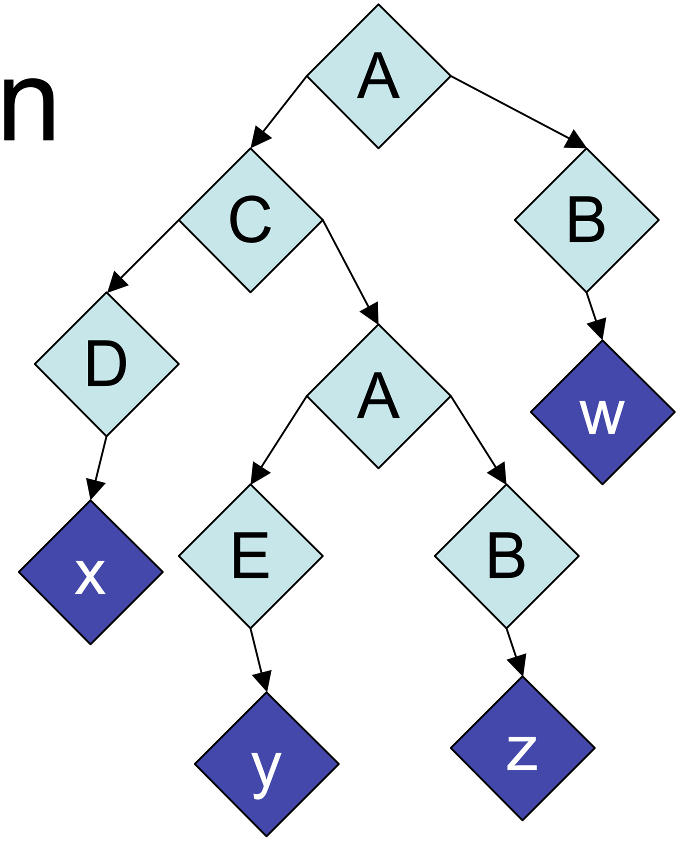
Illustration

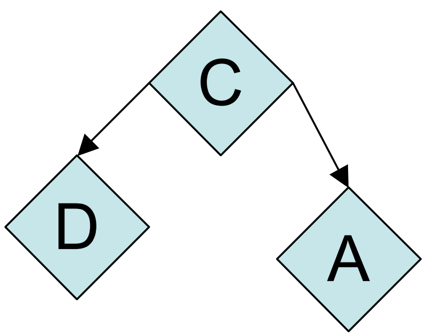


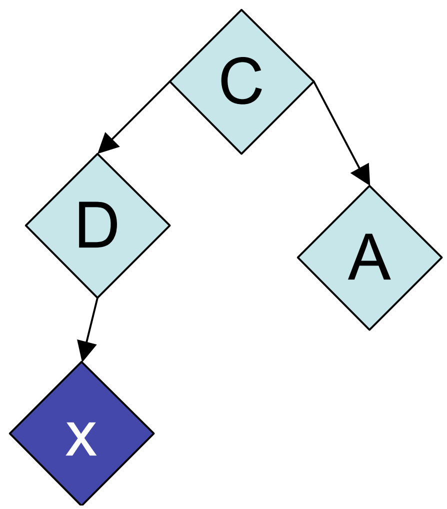


Illustration



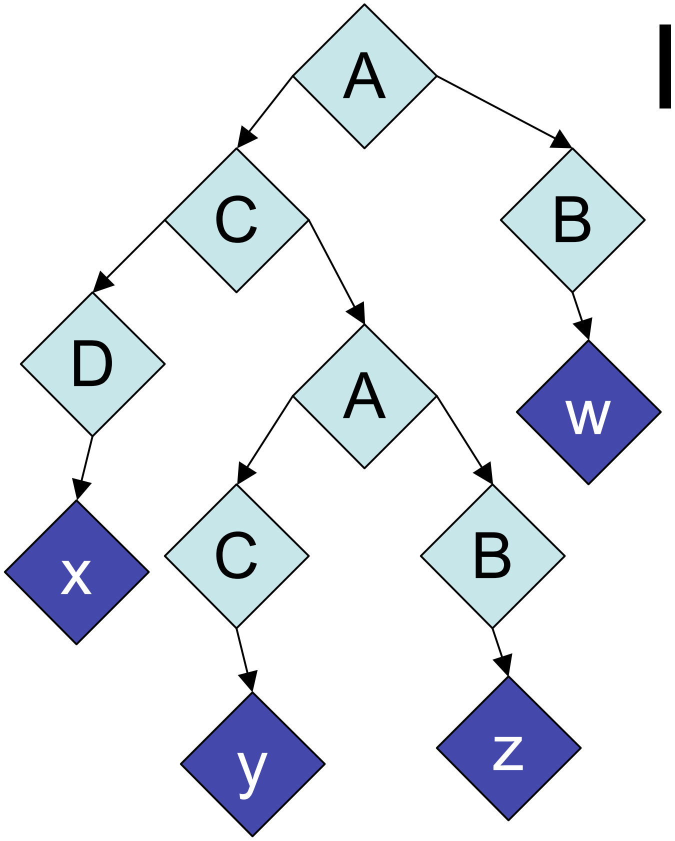


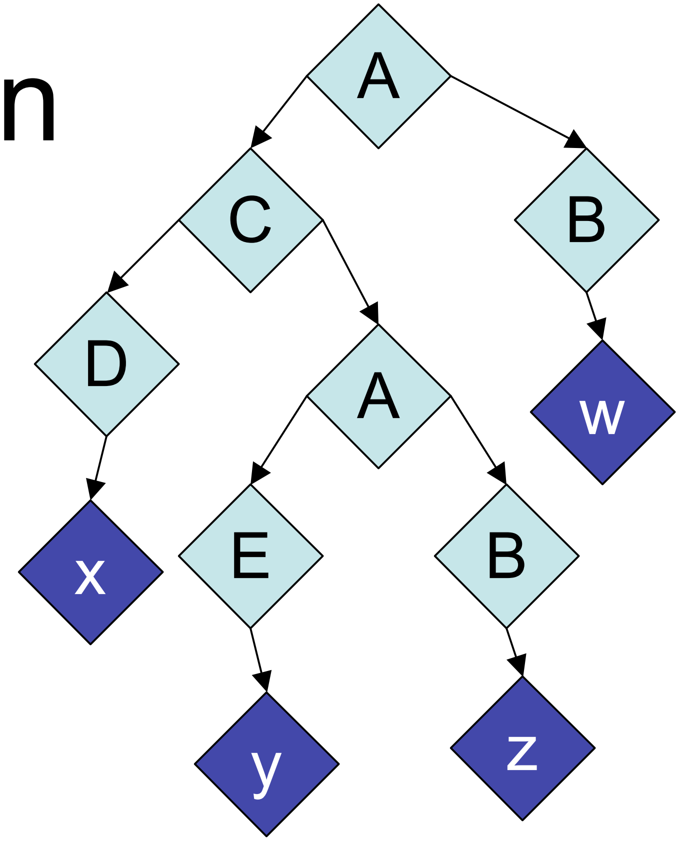


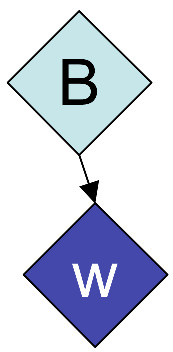


matches at (higher) C: 2

Illustration

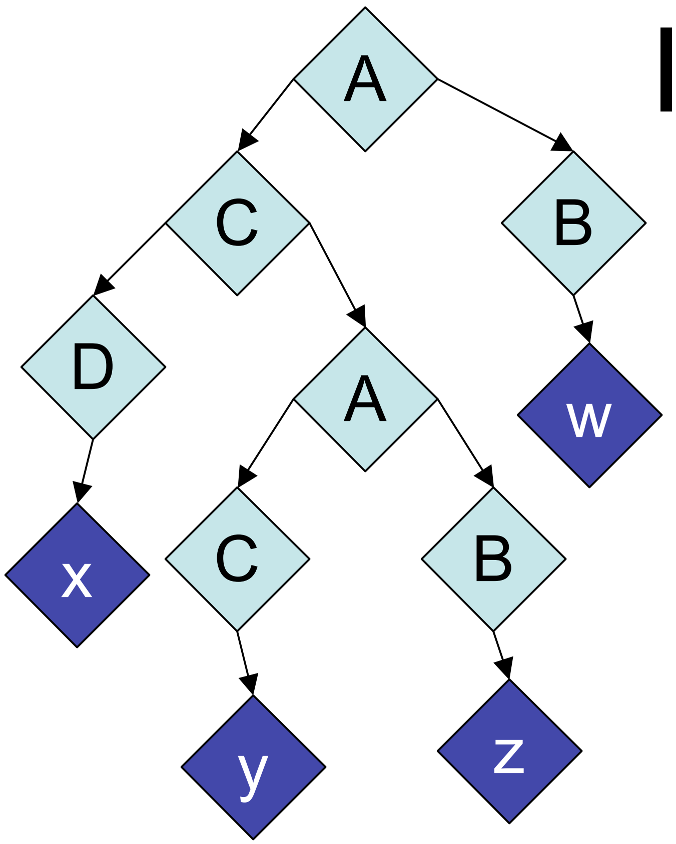


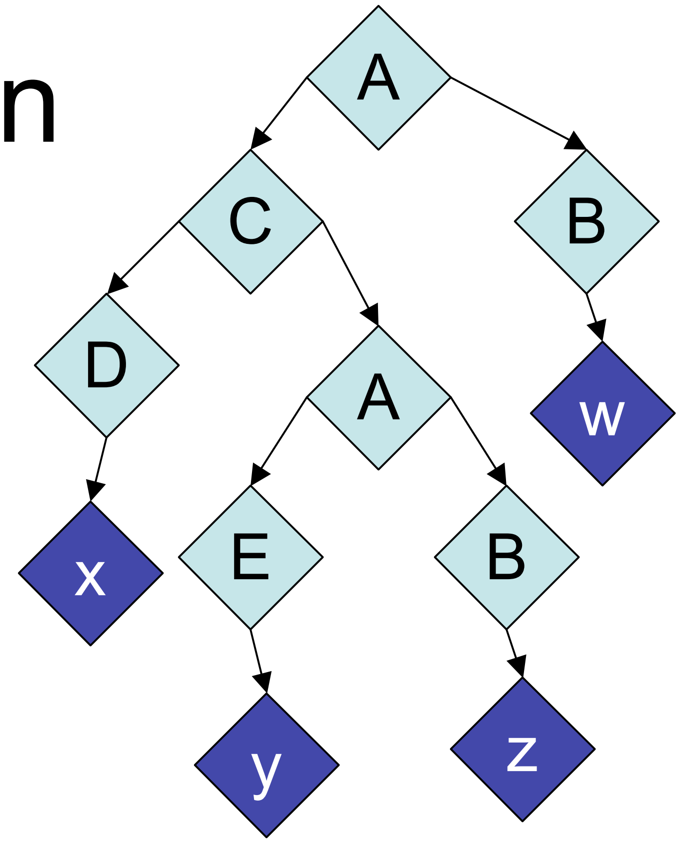




matches at (higher) C: 2
matches at (higher) B: 1

Illustration





matches at (higher) C: 2
matches at (higher) B: 1

$$\Delta(A_{1,4}, A_{1,4}) = (1 + \Delta(C_{1,3}, C_{1,3})) \times (1 + \Delta(B_{4,4}, B_{4,4})) = 6$$

NB

- Labeled sequences are trees, too.
  - As we saw!
- So you can define an “all-fragments” kernel for labeled sequences in **exactly the same way**.
- Try it!

Problem with the Dual Representation

- Decoding for parsing and tagging models usually involves dynamic programming (max-times).
  - For that, we need **w**.
  - Need to convert back to the **primal**.
- How many different α?
  - Exponential! (Size of $\Sigma_i \text{GEN}(x_i)$.)

$$\mathbf{w} = \sum_i \sum_{y' \in \text{GEN}(y_i)} \alpha_{i,y'} \left(\mathbf{f}\left(x_i, y'\right)\right) \qquad K\left(\mathbf{f}\left(x, y\right), \mathbf{w}\right) = \sum_i \sum_{y' \in \text{GEN}(y_i)} \alpha_{i,y'} K\left(\mathbf{f}\left(x, y\right), \mathbf{f}\left(x_i, y'\right)\right)$$

Discriminative Reranking

- Reduce the size of GEN.
- Use a base model to propose a list of the top *N* structures.
  - Usually done approximately until 2005.
  - Goodman (1999): *N*-best semiring (not efficient)
  - Huang and Chiang (2005): general solution to *N*-best lists for (max-plus) dynamic programming algorithms.
- Train a model to discriminate **correct** structure from other top-*N* structures.

Discriminative Reranking

- Collins (2000):
  - Exp-loss evaluated
  - Log-loss defined, not tested (more expensive)
- See Riezler et al. (2002) and Charniak and Johnson (2005) for log-loss results.
- Don’t need kernels for this!
  - Still unclear how easily we can mix kernels with log-loss.
- Great way to throw in features that are too expensive to put into a dynamic programming algorithm.

Everything That's Old is New Again

- Brill (1992): "transformation-based learning"
- No model and no weights.
- Transformation: a rule that modifies the structure.
  - E.g., "if the word is *dog* and the word before it is *the*, tag the word NOUN."
  - Think of these as "find-replace" operators.
- What is learned?
  - A sequence of deterministic transformations.
- Applied to tagging, NER, parsing, …
- Application: apply transformations in sequence.
  - Really fast!
- Training …

TBL Training

- Specify all rule templates.
- Apply baseline to training data.
- L = $\langle\rangle$
- Iterate until performance decreases on dev set:
  - Choose the rule R that increases net accuracy by the most (if it exists; else quit). (Expensive search!)
  - Add the rule to the end of L.
  - Apply R to the training data.
- Return L.

Notes:

This is a greedy learner.

Error rate on training data is **guaranteed** to decrease until termination.

Recap of Discriminative Methods: Attacking Error Directly

- Linear separation and the perceptron
- Conditional estimation, Boosting, Maximum Margin training
  - 0-1 loss, log-loss, exp-loss, hinge loss
- Lagrange duality
- Support vector machines
- Kernels
  - Tree kernels
- Reranking
- Transformation-based learning

Next Time …

Unsupervised learning: models gone wild!