

# Unsupervised learning as supervised structured prediction

**Hal Daumé III**

School of Computing  
University of Utah

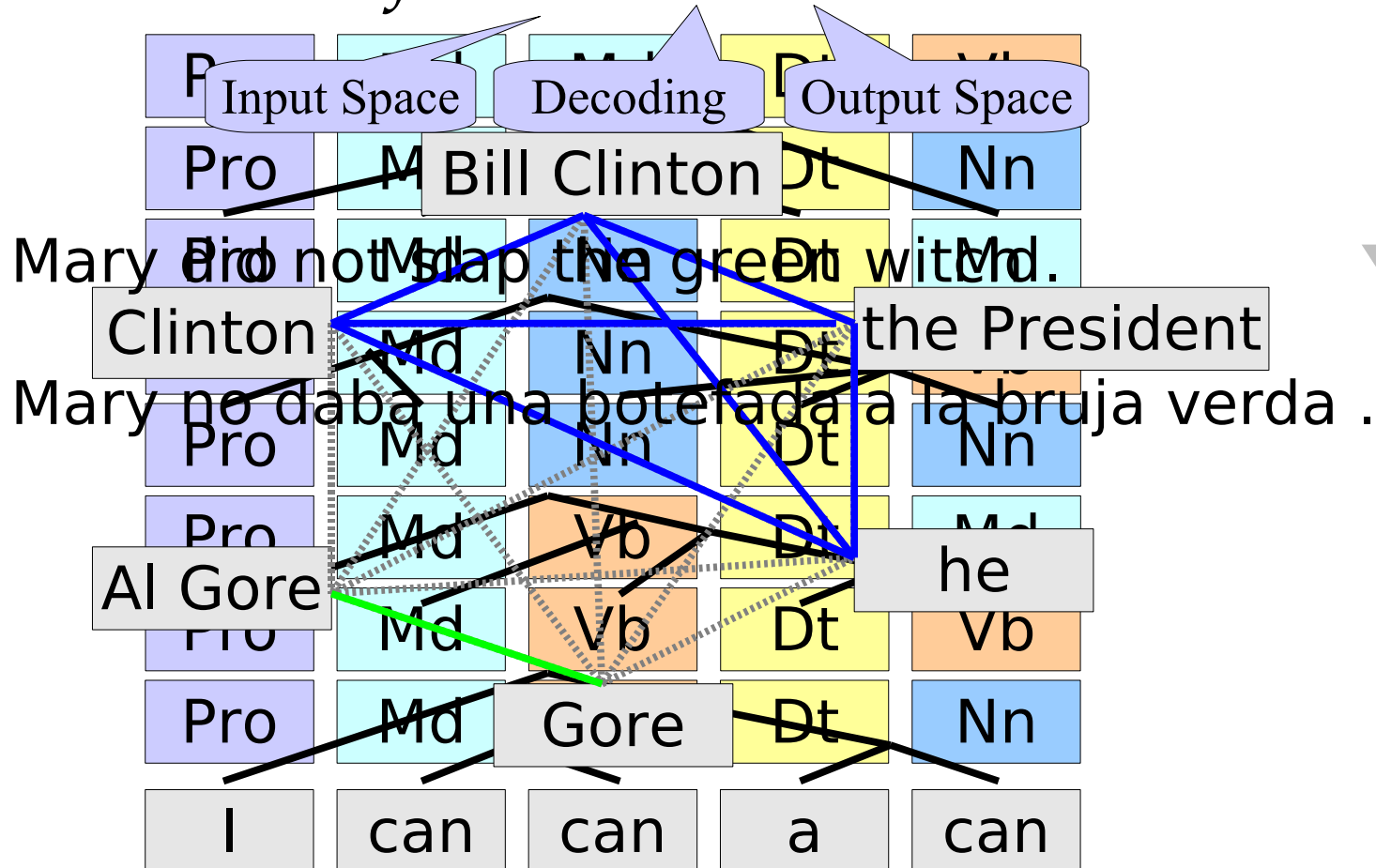
me@hal3.name



# Structured Prediction 101

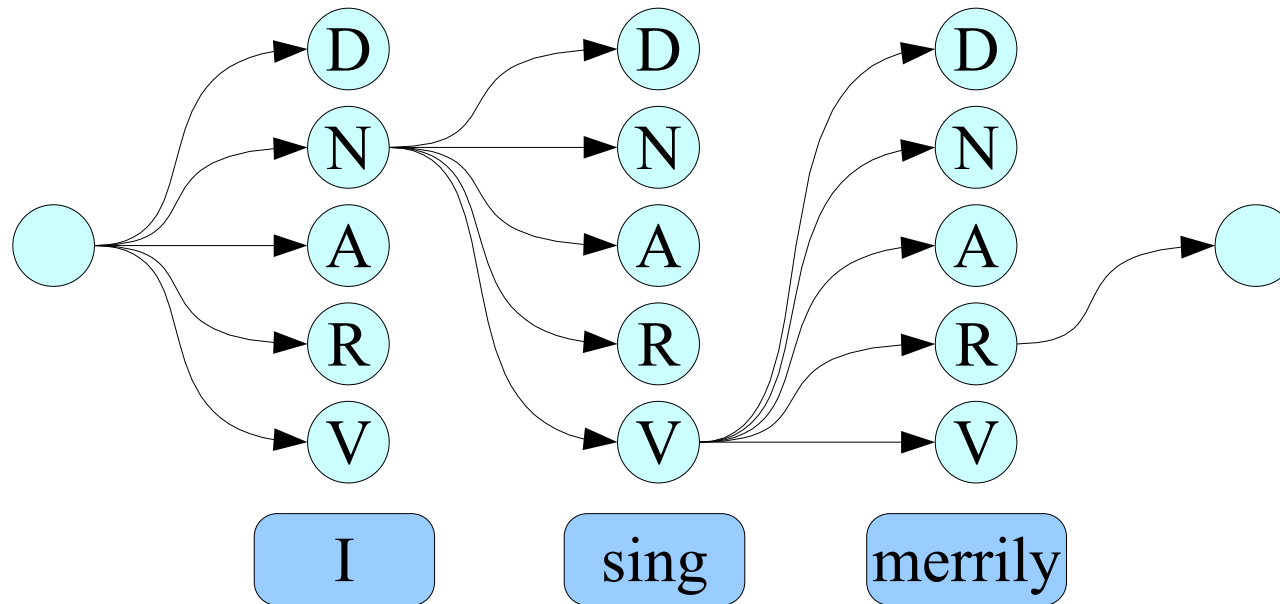
- Learn a function mapping inputs to complex outputs:

$$f : X \rightarrow Y$$



Mapping the Relation

# Reducing Structured Prediction



**Key Assumption:** *Optimal Policy for training data*

**Given:** input, true output and state;  
**Return:** best successor state

**Weak!**

# Theoretical Analysis

**Theorem:** For conservative  $\beta$ , after  $2T^3 \ln T$  iterations, the loss of the learned policy is bounded as follows:

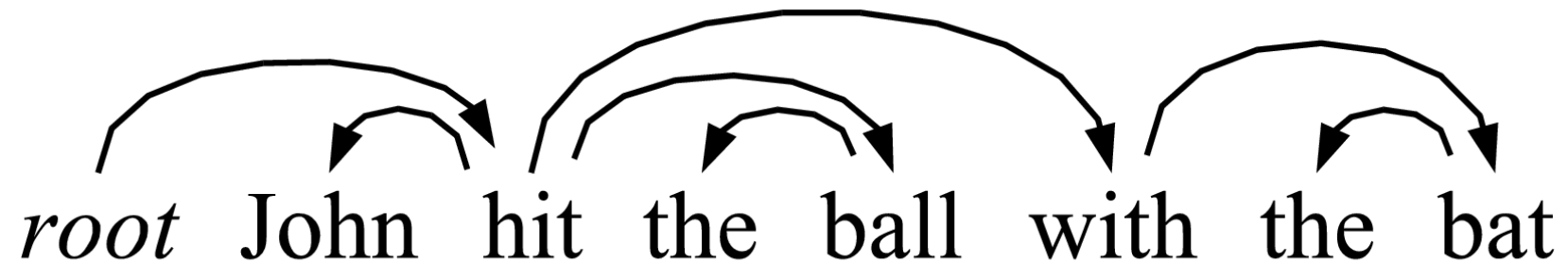
$$L(h) \leq L(h_0) + 2T \ln T l_{avg} + (1 + \ln T) \frac{c_{max}}{T}$$

Loss of the optimal policy

Average multiclass classification loss

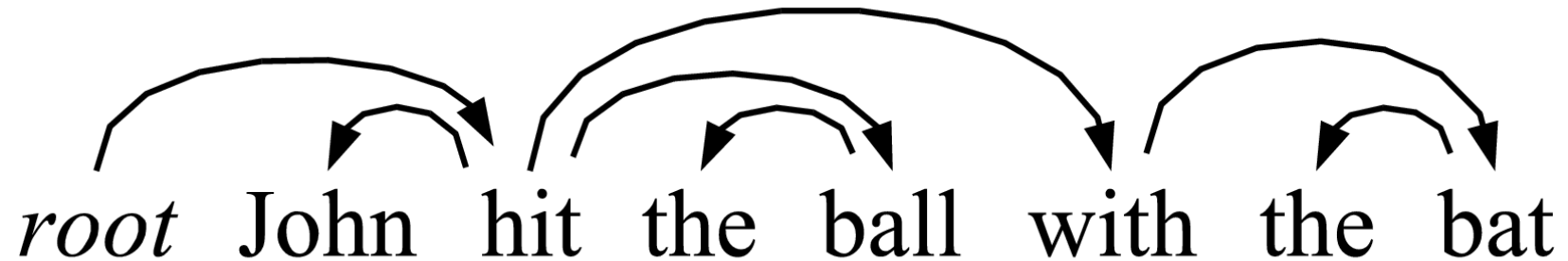
Worst case per-step loss

# Dependency Parsing



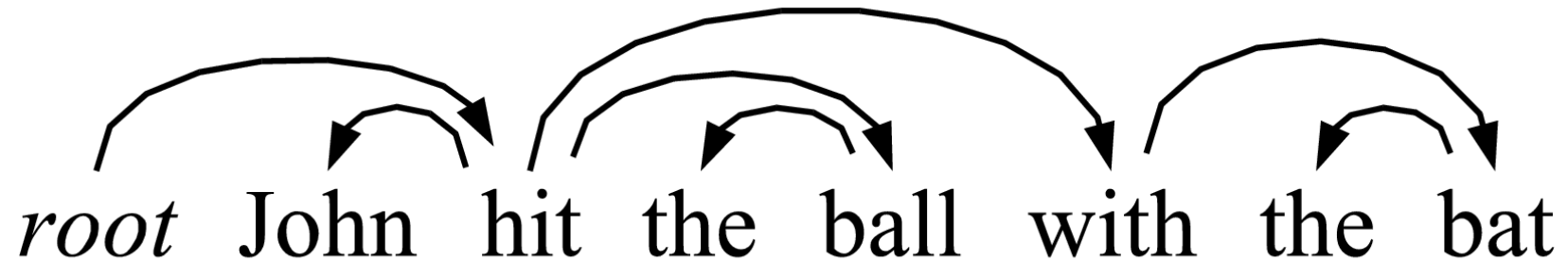
Nivre-style shift-reduce algorithm

# Dependency Parsing



- Start in state  $(S, i, A) = (\{\}, 1, \{\})$  and end when  $i=N$
- One of four actions:
  - LeftA:  $(t:S, i, A)$  to  $(S, i, (i,t):A)$   
so long as no  $(?,t)$  in  $A$
  - RightA:  $(t:s, i, A)$  to  $(i:t:S, i+1, (t,i):A)$   
so long as no  $(?,i)$  in  $A$
  - Reduce:  $(t:S, i, A)$  to  $(S, i, A)$  so long as  $(?,t)$  in  $A$
  - Shift:  $(S, i, A)$  to  $(i:S, i+1, A)$

# Unsupervised prediction



Key idea (not new!): a tree is good if it enables us to do a good job re-predicting the input

**eg., we should be able to predict “with” given its parent “hit” and dependent “bat”**

# Algorithm

- Initialize with random trees
- Repeat:
  - Train a classifier  $A$  to predict sentences given trees, optimized for some loss function  $L$
  - Train a classifier  $B$  to predict trees given sentences, optimized for  $L(A)$
  - Predict trees using  $B$
- In other words, a tree has low loss if it aids in the reproduction of the input

# Mini attempt at an analysis

- Under two key assumptions,  $L(A)$  will decrease with each iteration
- Assumption 1:
- Assumption 2:

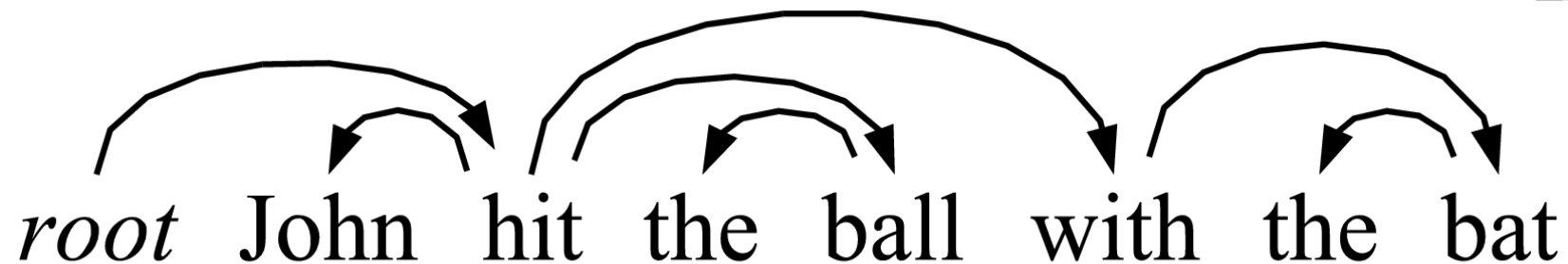
# Minor Aside

- In sequence labeling problems, this algorithm yields exactly the forward backward algorithm when:
  - The loss is Hamming loss
  - Naïve Bayes classifiers are used
  - Features are just tag/tag and tag/word pairs
  - Search losses are computed using dynamic programming

Model	States	Truth	EM	SEARN-NB	SEARN-LR
HMM1	$K = 2$	$0.227 \pm 0.107$	$0.275 \pm 0.128$	<b><math>0.287 \pm 0.138</math></b>	$0.276 \pm 0.095$
HMM1	$K = 5$	$0.687 \pm 0.043$	$0.678 \pm 0.026$	$0.688 \pm 0.025$	<b><math>0.672 \pm 0.022</math></b>
HMM1	$K = 10$	$0.806 \pm 0.035$	$0.762 \pm 0.021$	$0.771 \pm 0.019$	<b><math>0.755 \pm 0.019</math></b>
HMM2	$K = 2$	$0.294 \pm 0.072$	$0.396 \pm 0.057$	$0.408 \pm 0.056$	<b><math>0.271 \pm 0.057</math></b>
HMM2	$K = 5$	$0.651 \pm 0.068$	$0.695 \pm 0.027$	$0.710 \pm 0.016$	<b><math>0.633 \pm 0.018</math></b>
HMM2	$K = 10$	$0.815 \pm 0.032$	$0.764 \pm 0.021$	$0.771 \pm 0.015$	<b><math>0.705 \pm 0.019</math></b>

# Back to Parsing

Model	Accuracy-Train	Accuracy-Test	Iterations
Random-Generative	0.235 $\pm$ 0.009	0.235 $\pm$ 0.013	
Random-SEARN	0.213 $\pm$ 0.002	0.210 $\pm$ 0.006	
Klein+Manning:Random-Init	0.236 $\pm$ 0.038	0.236 $\pm$ 0.043	63.3 $\pm$ 9.2
Klein+Manning:Smart-Init	0.352 $\pm$ 0.066	0.352 $\pm$ 0.060	64.1 $\pm$ 11.1
Smith+Eisner:Length	0.338 $\pm$ 0.036	0.337 $\pm$ 0.059	173.1 $\pm$ 77.7
Smith+Eisner:Trans1	0.488 $\pm$ 0.009	0.490 $\pm$ 0.015	173.4 $\pm$ 71.0
Smith+Eisner:DelOrTrans1	0.473 $\pm$ 0.060	0.471 $\pm$ 0.059	132.2 $\pm$ 29.9
SEARN:Unsupervised	0.438 $\pm$ 0.016	0.434 $\pm$ 0.022	27.6 $\pm$ 3.7
Smith+Eisner:Supervised	0.799 $\pm$ 0.002	0.786 $\pm$ 0.008	350.5 $\pm$ 54.4
SEARN:Supervised	0.800 $\pm$ 0.003	0.806 $\pm$ 0.004	24.4 $\pm$ 2.6



# Discussion

- Proposed algorithm that naturally extends Searn to the unsupervised setting
- Key (non-novel) idea: predict the input
- Comes with (somewhat trivial) guarantees
  - Algorithmically resembles Viterbi EM, but will converge
  - Assumptions are, occasionally, verifiable
- Can train unsupervised sequence labelers with SVMs or DTs whatever classifier you want with whatever feature spaces you want
- Can do the same for any structured prediction problem
- Want: better understanding of why it works