Generative and Discriminative Models in Statistical Parsing

Michael Collins

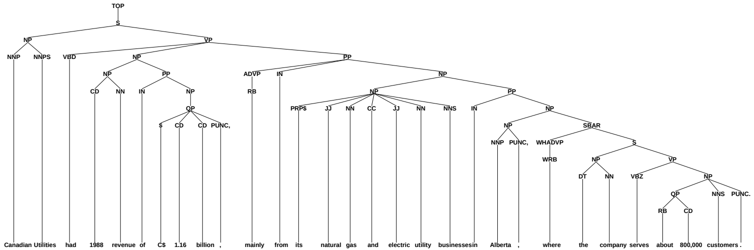
MIT

December 11, 2009

Parsing

Canadian Utilities had 1988 revenue of C$ 1.16 billion, mainly from its natural gas and electric utility businesses in Alberta, where the company serves about 800,000 customers.

$$\Downarrow$$



Outline

- Generative and discriminative models for parsing:
  - SPATTER
  - 5 lexicalized models
- Two hybrid generative/discriminative models

Discriminative Model 1: SPATTER

(Magerman 1995; Jelinek et al 1994)

- Input sentence $= x$, parse tree y represented as a sequence of decisions, $d_1 d_2 \ldots d_n$.

$$P(y|x) = \prod_{i=1}^{n} P(d_i | d_1 \ldots d_{i-1}, x)$$

$P(d_i | d_1 \ldots d_{i-1}, x)$ estimated using decision trees

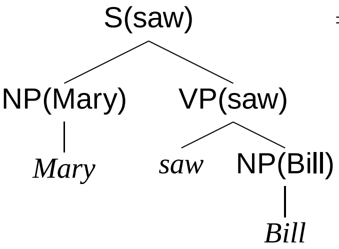
The Label-Bias Problem

$$P(y|x) = \prod_{i=1}^{n} P(d_i | d_1 \ldots d_{i-1}, x)$$

- If you think the label-bias problem is bad for MEMMs, you should try parsing...

NP	VP	NP	CC	N	V	N
|	|	|	|	|	|	|
N	VB	N	*and*	*jane*	*likes*	*bill*
|	|	|				
bill	*likes*	*mary*				

Discriminative Model 2: Lexical Dependencies (C, 1996)



$\Rightarrow$

Head	Modifier	Label
saw	Mary	S-VP-NP
saw	Bill	VP-V-NP
**	saw	ROOT

- The “probability” for this parse tree:

$$P(\text{S-VP-NP}|\text{Mary saw Bill}) \times P(\text{VP-V-NP}|\text{Mary saw Bill})$$
$$\times P(\text{ROOT}|\text{Mary saw Bill})$$

Results

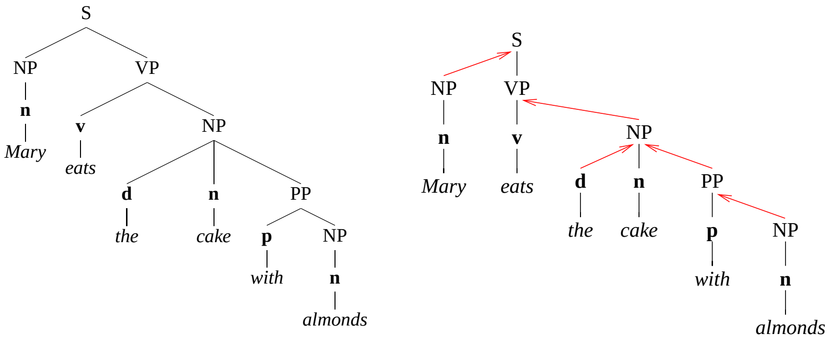
Model	F-measure
D1: SPATTER	**84.1**
D2	**85.5**
G1	87.8
G2	89.6
D4	91.1

- D1: $P(y|x) = \prod_{i=1}^{n} P(d_i|d_1 \ldots d_{i-1}, x)$
- D2: $P($S-VP-NP$|$Mary saw Bill$)$
- D2 gives some improvements, and is considerably simpler, *but* it's pretty suspect as a probabilistic model

Generative Models 1, 2: Markov Grammars

(C, 1997; Charniak, 1997/1999)

A parse tree is represented as a set of *spines* and *adjunctions*:



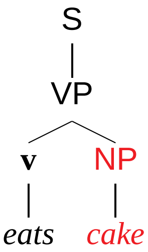
Markov Grammars (continued)

$$
\begin{array}{c}
\text{S} \\
| \\
\text{VP} \\
| \\
\mathbf{v} \\
| \\
\textit{eats}
\end{array}
$$

$$P(\text{S-VP-v-eats}|\text{ROOT})$$

- Each spine has a separate left/right weighted finite-state automaton (HMM) at each level of the tree (in this case S, VP)
- The automata generate sequences of modifier spines at each level of the tree

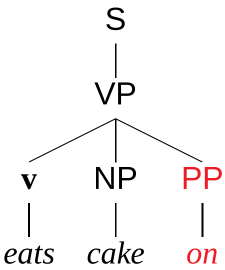
Markov Grammars (continued)



$$P(\text{NP-cake} \mid \text{VP-v-eats, RIGHT, ADJACENT})$$

- Each spine has a separate left/right weighted finite-state automaton (HMM) at each level of the tree (in this case S, VP)
- The automata generate sequences of modifier spines at each level of the tree

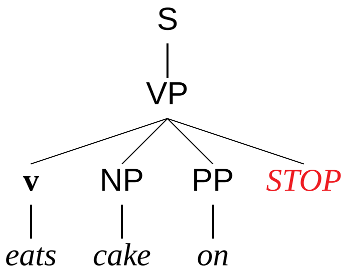
Markov Grammars (continued)



$$P(\text{PP-on}|\text{VP-v-eats, RIGHT, !ADJACENT})$$

- Each spine has a separate left/right weighted finite-state automaton (HMM) at each level of the tree (in this case S, VP)
- The automata generate sequences of modifier spines at each level of the tree

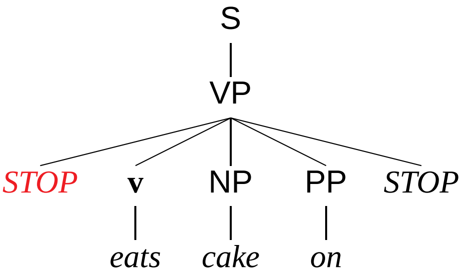
Markov Grammars (continued)



$$P(\text{STOP}|\text{VP-v-eats, RIGHT, !ADJACENT})$$

- Each spine has a separate left/right weighted finite-state automaton (HMM) at each level of the tree (in this case S, VP)
- The automata generate sequences of modifier spines at each level of the tree

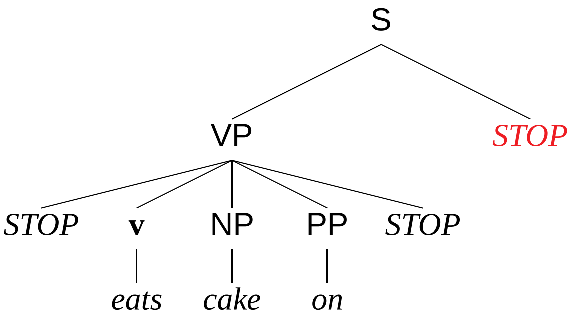
Markov Grammars (continued)



- Each spine has a separate left/right weighted finite-state automaton (HMM) at each level of the tree (in this case S, VP)

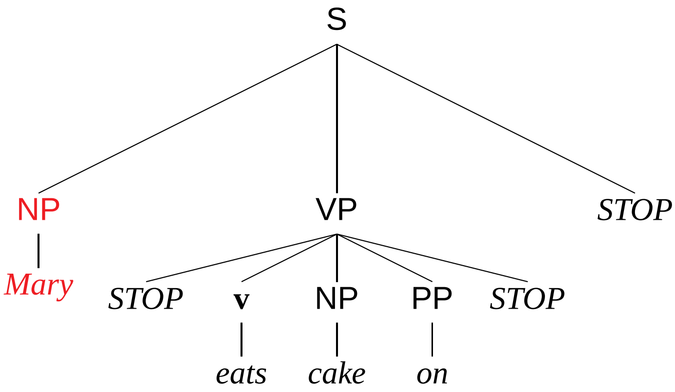
- The automata generate sequences of modifier spines at each level of the tree

Markov Grammars (continued)



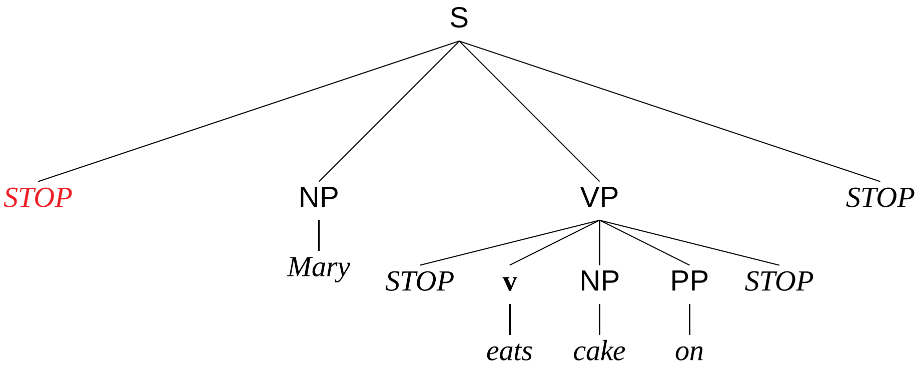
- Each spine has a separate left/right weighted finite-state automaton (HMM) at each level of the tree (in this case S, VP)
- The automata generate sequences of modifier spines at each level of the tree

Markov Grammars (continued)



- Each spine has a separate left/right weighted finite-state automaton (HMM) at each level of the tree (in this case S, VP)
- The automata generate sequences of modifier spines at each level of the tree

Markov Grammars (continued)



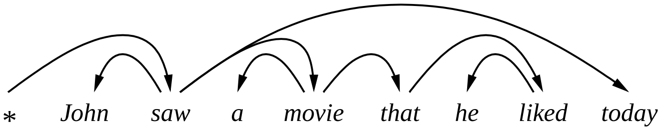
- Each spine has a separate left/right weighted finite-state automaton (HMM) at each level of the tree (in this case S, VP)
- The automata generate sequences of modifier spines at each level of the tree

Results

Model	F-measure
D1: SPATTER	84.1
D2	**85.5**
G1	**87.8**
G2	**89.6**
D4	91.1

- D2: $P(\text{S-VP-NP}|\text{Mary saw Bill})$
- G1/G2: $P(\text{NP-cake}|\text{VP-v-eats, RIGHT, ADJACENT, } \ldots)$
- Markov grammars are coherent probabilistic models, and give improvements, but there are many details...

Discriminative Model 3: (McDonald et al, 2005)



- A discriminative model for dependency parsing:

$$\mathbf{y}^* = \arg\max_{\mathbf{y}} \sum_{r \in \mathbf{y}} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, r)$$

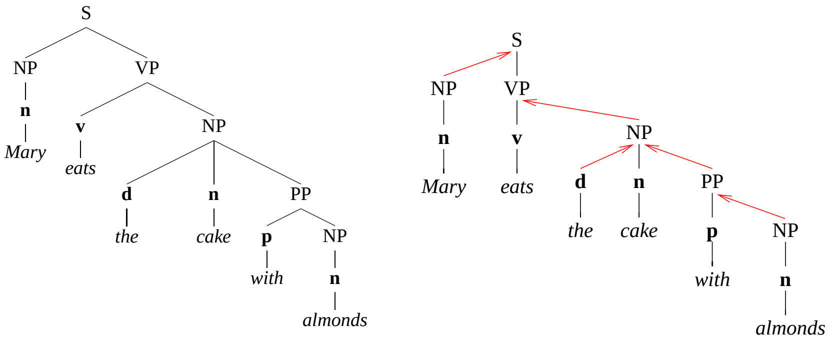
  where each r is a tuple $\langle h, m \rangle$ representing a dependency from modifier m to head h

- $\mathbf{f}(\mathbf{x}, r)$ is a feature vector associated with dependency r, $\mathbf{w}$ is a parameter vector (trained using MIRA, averaged perceptron, etc.)
- A simple, direct model, allows easy incorporation of features. **Very easy to replicate**

Discriminative Model 4: a TAG-Based Model

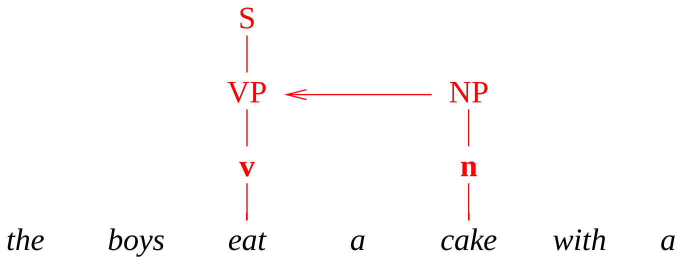
(Carreras, C, and Koo, 2008)

A parse tree is represented as a set of *spines* and *adjunctions*:



Discriminative Model 4: a TAG-Based Model

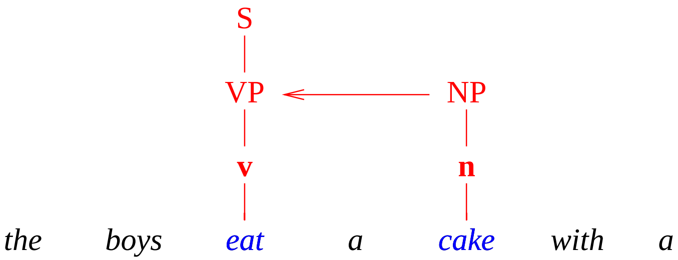
(Carreras, C, and Koo, 2008)



- Feature vectors $\mathbf{f}(\mathbf{x}, h, m, \sigma_h, \sigma_m, \text{POS})$ where
  - $\mathbf{x}$ is the sentence
  - $h = 3$ (index of head word), $m = 5$ (index of modifier word)
  - σ_h and σ_m are the head and modifier spines
  - POS is the position being adjoined into (e.g., VP)

Discriminative Model 4: a TAG-Based Model

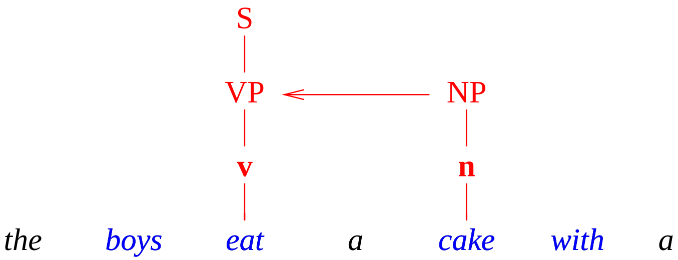
(Carreras, C, and Koo, 2008)



- Feature vectors $\mathbf{f}(\mathbf{x}, h, m, \sigma_h, \sigma_m, \text{POS})$ where
  - $\mathbf{x}$ is the sentence
  - $h = 3$ (index of head word), $m = 5$ (index of modifier word)
  - σ_h and σ_m are the head and modifier spines
  - POS is the position being adjoined into (e.g., VP)

Discriminative Model 4: a TAG-Based Model

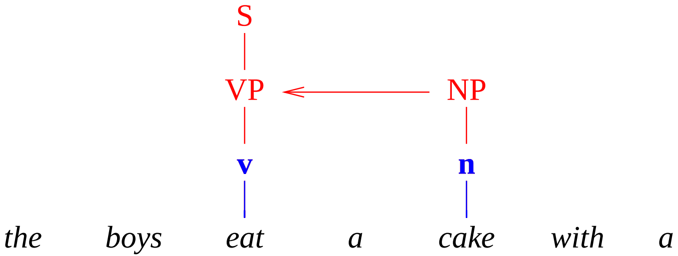
(Carreras, C, and Koo, 2008)



- Feature vectors $\mathbf{f}(\mathbf{x}, h, m, \sigma_h, \sigma_m, \text{POS})$ where
  - $\mathbf{x}$ is the sentence
  - $h = 3$ (index of head word), $m = 5$ (index of modifier word)
  - σ_h and σ_m are the head and modifier spines
  - POS is the position being adjoined into (e.g., VP)

Discriminative Model 4: a TAG-Based Model

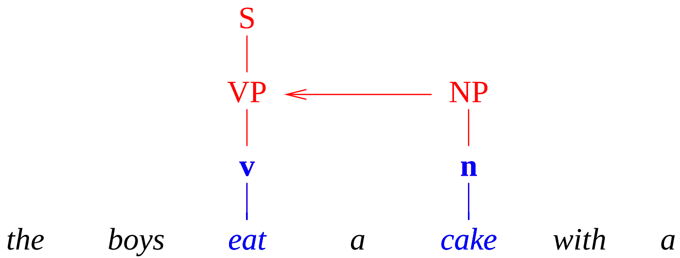
(Carreras, C, and Koo, 2008)



- Feature vectors $\mathbf{f}(\mathbf{x}, h, m, \sigma_h, \sigma_m, \text{POS})$ where
  - $\mathbf{x}$ is the sentence
  - $h = 3$ (index of head word), $m = 5$ (index of modifier word)
  - σ_h and σ_m are the head and modifier spines
  - POS is the position being adjoined into (e.g., VP)

Discriminative Model 4: a TAG-Based Model

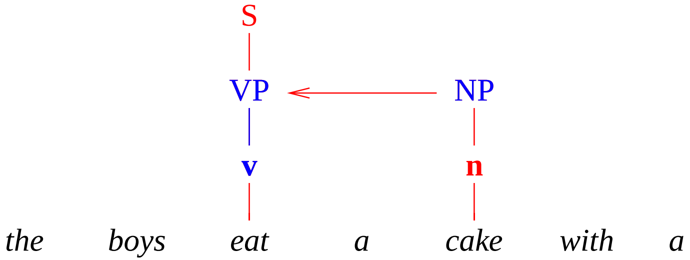
(Carreras, C, and Koo, 2008)



- Feature vectors $\mathbf{f}(\mathbf{x}, h, m, \sigma_h, \sigma_m, \text{POS})$ where
  - $\mathbf{x}$ is the sentence
  - $h = 3$ (index of head word), $m = 5$ (index of modifier word)
  - σ_h and σ_m are the head and modifier spines
  - POS is the position being adjoined into (e.g., VP)

Discriminative Model 4: a TAG-Based Model

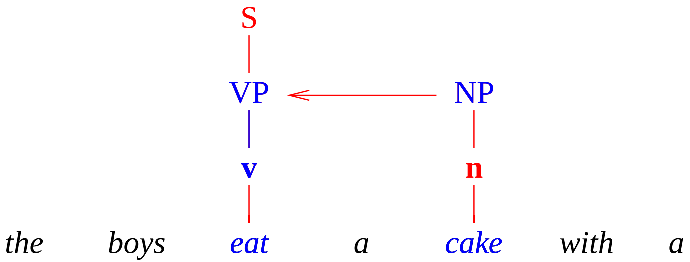
(Carreras, C, and Koo, 2008)



- Feature vectors $\mathbf{f}(\mathbf{x}, h, m, \sigma_h, \sigma_m, \text{POS})$ where
  - $\mathbf{x}$ is the sentence
  - $h = 3$ (index of head word), $m = 5$ (index of modifier word)
  - σ_h and σ_m are the head and modifier spines
  - POS is the position being adjoined into (e.g., VP)

Discriminative Model 4: a TAG-Based Model

(Carreras, C, and Koo, 2008)

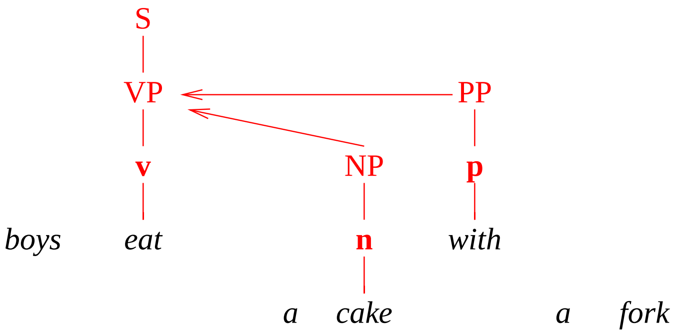


- Feature vectors $\mathbf{f}(\mathbf{x}, h, m, \sigma_h, \sigma_m, \text{POS})$ where
  - $\mathbf{x}$ is the sentence
  - $h = 3$ (index of head word), $m = 5$ (index of modifier word)
  - σ_h and σ_m are the head and modifier spines
  - POS is the position being adjoined into (e.g., VP)

Discriminative Model 4: a TAG-Based Model

(Carreras, C, and Koo, 2008)

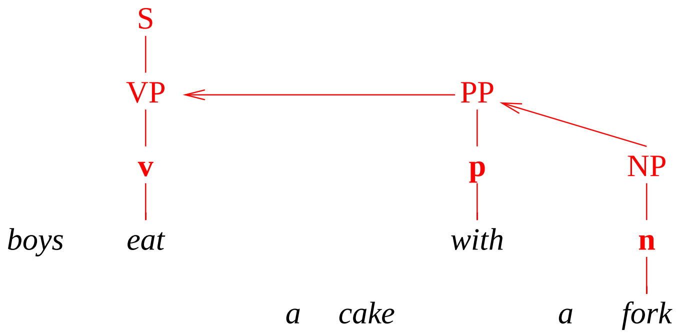
Trigram dependency features:



Discriminative Model 4: a TAG-Based Model

(Carreras, C, and Koo, 2008)

More trigram dependency features:



Results

Model	F-measure
D1: SPATTER	**84.1**
D2	85.5
G1	87.8
G2	89.6
D4	**91.1**

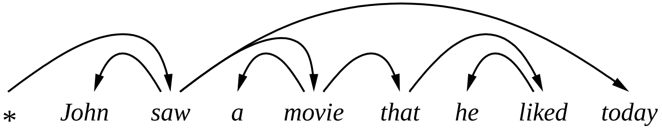
- D1:

$$y^* = \arg\max_y \sum_{i=1}^{n} \log P(d_i | d_1 \dots d_{i-1}, x)$$

- D4:

$$y^* = \arg\max_y \sum_{r \in y} \mathbf{w} \cdot \mathbf{f}(x, r)$$

"Hybrid" Discriminative/Generative Model 1: Word Clusters (Koo, C, Carreras, 2008)



$$\mathbf{y}^* = \arg\max_{\mathbf{y}} \sum_{(h,m)\in\mathbf{y}} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, h, m)$$

- Feature vectors $\mathbf{f}(\mathbf{x}, h, m)$ depend heavily on lexical items, which are *sparse*
- A semi-supervised method: use unlabeled data to induce hierarchical word clusters, then use these within features

Results

Dependency accuracy for a 2nd order parser:

Training size	Baseline	Clusters	Improvement
1k	81.95	85.33	3.38
2k	85.02	87.54	2.52
4k	87.88	89.67	1.79
8k	89.71	91.37	1.66
16k	91.14	92.22	1.08
32k	92.09	93.21	1.12
All	92.42	93.30	0.88

"Hybrid" Discriminative/Generative Model 2

(Suzuki et al, 2009)

Step 1 Train a CRF-style dependency model on the labeled examples

$$y^* = \arg\max_y \sum_{r \in y} \mathbf{w} \cdot \mathbf{f}(x, r)$$

Step 2 Use the model from step 1 to produce parse trees on unlabeled data, and estimate generative models

$$P(y, x; \theta_i) \quad \text{for } i = 1 \ldots k$$

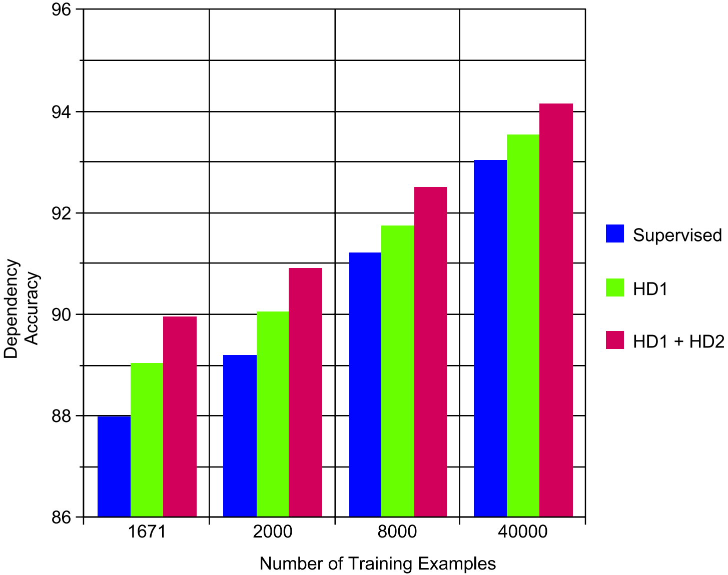
(typically $k \approx 100$)

Step 3 Add new features $\log P(y, x; \theta_i)$ for $i = 1 \ldots k$ to the supervised model, and retrain

The Generative Models

- The k generative models are derived directly from the original feature vectors $\mathbf{f}(x, r)$!
- First partition the feature vector into k sets of disjoint features (typically by feature type)
- Next, define a naive-bayes model for each partition

Results


96
94
92
90
88
86
Dependency Accuracy
1671
2000
8000
40000
Number of Training Examples
Supervised
HD1
HD1 + HD2

Final Thoughts

- Advantages of generative models:
  - Very fast to train
  - Very useful in semi-supervised approaches
  - Invaluable as language models in speech recognition, machine translation
  - Better than discriminative models with small amounts of training data? (I'm skeptical about this...)
- Advantages of discriminative models:
  - Very easy to incorporate new features (including features induced from unlabeled data)
  - Easy to implement and replicate (no issues of smoothing, independence assumptions etc. — all you need is the feature definitions)