# Bayesian Models for Dependency Parsing Using Pitman-Yor Priors

Hanna M. Wallach

University of Massachusetts Amherst
wallach@cs.umass.edu

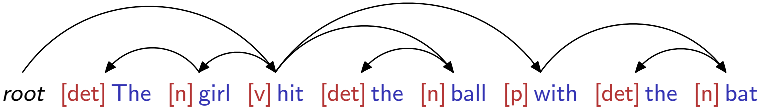
July 8, 2009

Joint work with Charles Sutton and Andrew McCallum

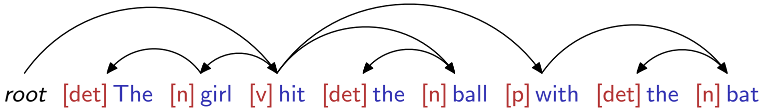
# Dependency Parsing

[det] The [n] girl [v] hit [det] the [n] ball [p] with [det] the [n] bat

# Dependency Parsing



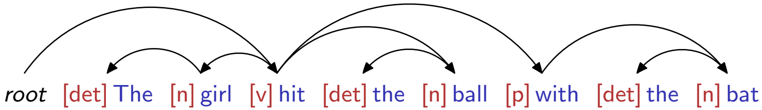
# Dependency Parsing




- Dependency trees encodes syntactic relationships between words
- Each node is a part-of-speech-tagged, cased[1] word
- An edge from word $w_m$ to $w_{m'}$ means $w_{m'}$ is a *dependent* of $w_m$

[1] Cases: upper, lower, mixed, first capitalized word

# Talk Outline




- Four hierarchical Bayesian dependency models:
  - Bayesian reinterpretation of a classic dependency model
  - Extension of this model using hierarchical Pitman-Yor priors
  - Bayesian dependency model with "syntactic" states
  - Bayesian dependency model with "semantic" states

# Eisner's Generative Dependency Model (Eisner '96)

- Conditioned on their parent, left and right children each form a first-order Markov chain
- Final child in each direction is a special *stop* symbol
- *Stop* symbols enable simultaneous generation of words and trees

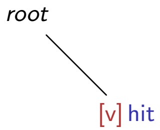
# Eisner's Generative Dependency Model (Eisner '96)

- Conditioned on their parent, left and right children each form a first-order Markov chain
- Final child in each direction is a special *stop* symbol
- *Stop* symbols enable simultaneous generation of words and trees

*root*

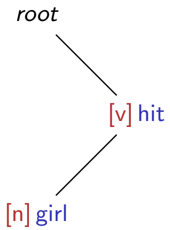
# Eisner's Generative Dependency Model (Eisner '96)

- Conditioned on their parent, left and right children each form a first-order Markov chain
- Final child in each direction is a special *stop* symbol
- *Stop* symbols enable simultaneous generation of words and trees



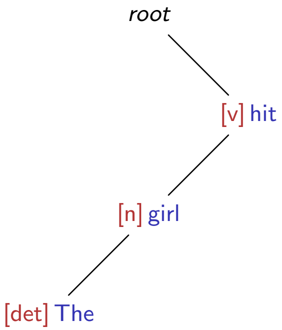
# Eisner's Generative Dependency Model (Eisner '96)

- Conditioned on their parent, left and right children each form a first-order Markov chain
- Final child in each direction is a special *stop* symbol
- *Stop* symbols enable simultaneous generation of words and trees



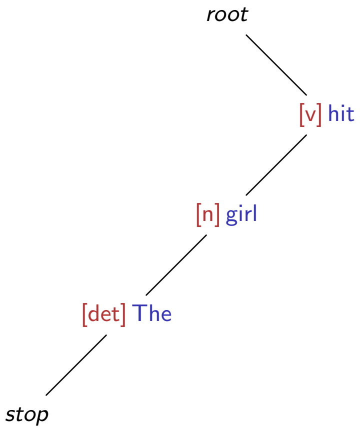
# Eisner's Generative Dependency Model (Eisner '96)

- Conditioned on their parent, left and right children each form a first-order Markov chain
- Final child in each direction is a special *stop* symbol
- *Stop* symbols enable simultaneous generation of words and trees



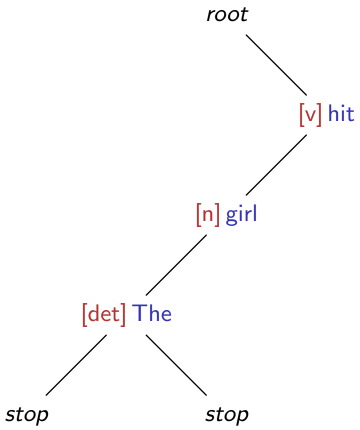
# Eisner's Generative Dependency Model (Eisner '96)

- Conditioned on their parent, left and right children each form a first-order Markov chain
- Final child in each direction is a special *stop* symbol
- *Stop* symbols enable simultaneous generation of words and trees



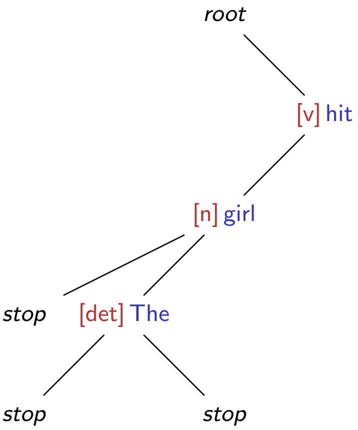
# Eisner's Generative Dependency Model (Eisner '96)

- Conditioned on their parent, left and right children each form a first-order Markov chain
- Final child in each direction is a special *stop* symbol
- *Stop* symbols enable simultaneous generation of words and trees



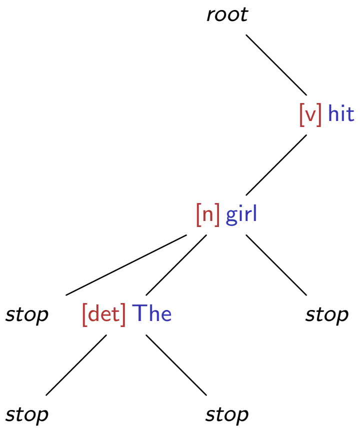
# Eisner's Generative Dependency Model (Eisner '96)

- Conditioned on their parent, left and right children each form a first-order Markov chain
- Final child in each direction is a special *stop* symbol
- *Stop* symbols enable simultaneous generation of words and trees



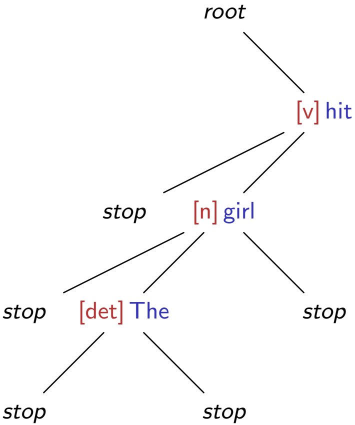
# Eisner's Generative Dependency Model (Eisner '96)

- Conditioned on their parent, left and right children each form a first-order Markov chain
- Final child in each direction is a special *stop* symbol
- *Stop* symbols enable simultaneous generation of words and trees



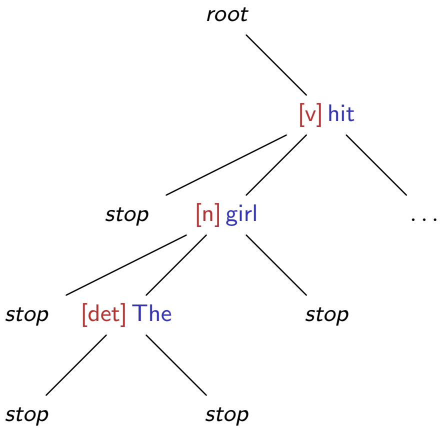
# Eisner's Generative Dependency Model (Eisner '96)

- Conditioned on their parent, left and right children each form a first-order Markov chain
- Final child in each direction is a special *stop* symbol
- *Stop* symbols enable simultaneous generation of words and trees

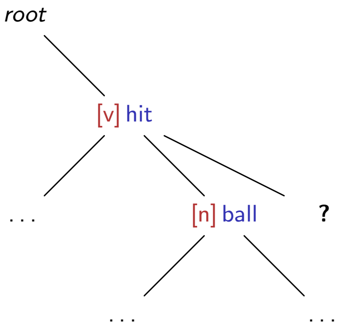


# Eisner's Generative Dependency Model (Eisner '96)

- Conditioned on their parent, left and right children each form a first-order Markov chain
- Final child in each direction is a special *stop* symbol
- *Stop* symbols enable simultaneous generation of words and trees

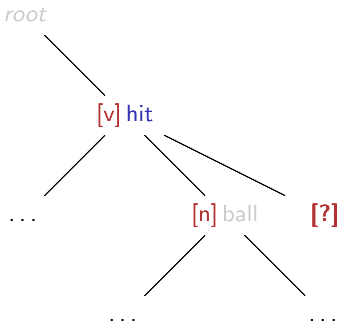


# Generating a Tagged, Cased Word



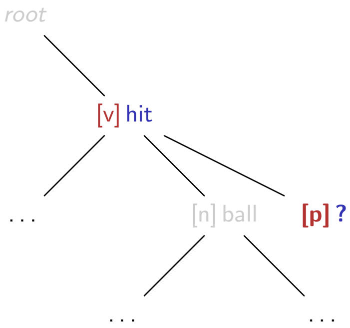
# Generating a Tagged, Cased Word

- Generate a **tag** given the tagged, cased parent word and the sibling tag



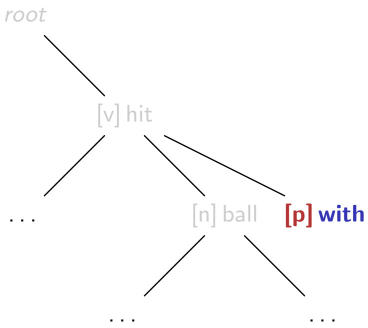
# Generating a Tagged, Cased Word

- Generate a **tag** given the tagged, cased parent word and the sibling tag
- Generate an **uncased word** given the tagged, cased parent word and the just-generated tag



# Generating a Tagged, Cased Word

- Generate a **tag** given the tagged, cased parent word and the sibling tag
- Generate an **uncased word** given the tagged, cased parent word and the just-generated tag
- Generate a case value given the just-generated tag and uncased word



# Estimating Probabilities from Data

- Use a corpus $\mathcal{D} = \{\mathbf{s}, \mathbf{w}, \mathbf{c}, \mathbf{t}\}$ of tagged, cased sentences and trees
- Count relevant occurrences in $\mathcal{D}$, e.g.,

$$N_{c|sw} = \# \text{ times uncased word } w \text{ with tag } s \text{ has case value } c$$

- Use counts to form "estimators", e.g., $N_{c|sw} \,/\, N_{\cdot|sw}$
- The more specific the context, the sparser the counts
- "Smooth" more specific estimators with less specific ones
- Same approach as interpolated $n$-gram language modeling

# Estimating Probabilities from Data

- Contexts are obvious in language modeling ($w_{n-2}\, w_{n-1} \longrightarrow w_{n-1}$)
- Choice of contexts is much less obvious in parsing, e.g.,

tag word $\longrightarrow$ tag   or   tag word $\longrightarrow$ word

- Eisner estimates e.g., the case value probability as follows:

$$P(\text{case} = c \,|\, \text{tag} = s, \text{word} = w, \mathcal{D}) = \frac{N_{c|sw} + 3\, \frac{N_{c|s} + 0.5\, \frac{1}{C}}{N_{\cdot|s} + 0.5}}{N_{\cdot|sw} + 3}$$

# Estimating Probabilities from Data

- Contexts are obvious in language modeling ($w_{n-2}\, w_{n-1} \;\longrightarrow\; w_{n-1}$)
- Choice of contexts is much less obvious in parsing, e.g.,

tag word $\longrightarrow$ tag  or  tag word $\longrightarrow$ word

- Eisner estimates e.g., the case value probability as follows:

$$P(\text{case} = c \,|\, \text{tag} = s, \text{word} = w, \mathcal{D}) = \frac{N_{c|sw} + 3\, \frac{N_{c|s} + 0.5\, \frac{1}{C}}{N_{\cdot|s} + 0.5}}{N_{\cdot|sw} + 3}$$

# Estimating Probabilities from Data

- Contexts are obvious in language modeling ($w_{n-2}\, w_{n-1} \;\longrightarrow\; w_{n-1}$)
- Choice of contexts is much less obvious in parsing, e.g.,

$$\text{tag word} \longrightarrow \text{tag} \quad \text{or} \quad \text{tag word} \longrightarrow \text{word}$$

- Eisner estimates e.g., the case value probability as follows:

$$P(\text{case} = c \,|\, \text{tag} = s, \text{word} = w, \mathcal{D}) = \frac{N_{c|sw} + 3\,\dfrac{N_{c|s} + 0.5\,\frac{1}{C}}{N_{\cdot|s} + 0.5}}{N_{\cdot|sw} + 3}$$

# Estimating Probabilities from Data

- Contexts are obvious in language modeling ($w_{n-2}\, w_{n-1} \;\rightarrow\; w_{n-1}$)
- Choice of contexts is much less obvious in parsing, e.g.,

tag word $\rightarrow$ tag or tag word $\rightarrow$ word

- Eisner estimates e.g., the case value probability as follows:

$$P(\text{case} = c \,|\, \text{tag} = s, \text{word} = w, \mathcal{D}) = \frac{N_{c|sw} + 3\, \dfrac{N_{c|s} + 0.5\, \frac{1}{C}}{N_{\cdot|s} + 0.5}}{N_{\cdot|sw} + 3}$$

# Contexts for Tags and Uncased Words

| $P(\text{tag} \mid \text{parent tagged cased word}, \text{sibling tag}, \text{dir})$ | | | | |
|---|---|---|---|---|
| parent tag | parent word | parent case | sibling tag | dir |
| parent tag | | | sibling tag | dir |
| parent tag | | | | dir |

| $P(\text{word} \mid \text{parent tagged cased word}, \text{dir})$ | | | | |
|---|---|---|---|---|
| tag | parent tag | parent word | parent case | dir |
| tag | parent tag | | | dir |
| tag | | | | |

# A Hierarchical Bayesian Dependency Model

- We can redefine Eisner's model from a Bayesian perspective
- Treat each probability vector as a random variable, e.g.,

$$\boldsymbol{\psi}_{sw} = \text{distribution over case values given context } s \text{ and } w$$

- Draw each probability vector from a Dirichlet prior, e.g.,

$$\boldsymbol{\psi}_{sw} \sim \mathrm{Dir}\,(\boldsymbol{\psi}_{sw};\, \alpha_1, \mathbf{m}_s)$$

- $\mathbf{m}_s$ is a tag-specific base measure (distribution over case values)

# Base Measures

- Base measures of Dirichlet priors, e.g., $\{\mathbf{m}_s\}_{s=1}^{S}$, are also unknown
- Can also draw each $\mathbf{m}_s$ from a Dirichlet prior
- Eisner: tag word $\rightarrow$ tag $\rightarrow$ uniform, so

$$\mathbf{m}_s \sim \mathrm{Dir}\left(\mathbf{m}_s;\, \alpha_0, \mathbf{u}\right)$$

- This induces a *hierarchical* Dirichlet prior over $\boldsymbol{\psi}_{sw}$
- Can integrate out $\mathbf{m}_s$ and $\boldsymbol{\psi}_{sw}$ to obtain the *predictive distribution*

# Predictive Distributions

- Predictive probability of case value $c$ is

$$P(\text{case} = c \,|\, \text{tag} = s, \text{word} = w, \mathcal{D}, \alpha_1, \alpha_0) =$$
$$\frac{N_{c|sw} + \alpha_1 \, \frac{\hat{N}_{c|s} + \alpha_0 \, \frac{1}{C}}{\hat{N}_{\cdot|s} + \alpha_0}}{N_{\cdot|sw} + \alpha_1}$$

- Bottom-level counts $N_{c|sw}$ and $N_{\cdot|sw}$ are raw observation counts
- Higher-level counts are not necessarily raw observation counts

# Relationship to Eisner's Model

- Compare Eisner's probabilities with predictive distributions, e.g.,

**Eisner**

$$P(c \mid s, w, \mathcal{D}) = \frac{N_{c|sw} + 3 \, \frac{N_{c|s} + 0.5 \, \frac{1}{C}}{N_{\cdot|s} + 0.5}}{N_{\cdot|sw} + 3}$$

**Bayesian**

$$P(c \mid s, w, \mathcal{D}, \alpha_1, \alpha_0) = \frac{N_{c|sw} + \alpha_1 \, \frac{\hat{N}_{c|s} + \alpha_0 \, \frac{1}{C}}{\hat{N}_{\cdot|s} + \alpha_0}}{N_{\cdot|sw} + \alpha_1}$$

- Only differences: concentration parameters, higher-level counts

# Advantages of the Bayesian Reinterpretation

- There are (at least) three ways of varying the Bayesian model:
  1. Concentration parameters (e.g., $\alpha_1$, $\alpha_0$) can be sampled
     - Need not be arbitrarily chosen or set using cross validation
  2. Counts need not correspond to observation counts
  3. Can use priors other than the hierarchical Dirichlet distribution
     - e.g., the hierarchical Pitman-Yor process
- All three variations have the potential to improve model quality

# Using Hierarchical Pitman-Yor Priors

- Can use Pitman-Yor priors instead of Dirichlet priors, e.g.,

$$\boldsymbol{\psi}_{sw} \sim \mathrm{PY}\left(\boldsymbol{\psi}_{sw} \mid \alpha_1, \epsilon_1, \mathbf{m}_s\right)$$
$$\mathbf{m}_s \sim \mathrm{PY}\left(\mathbf{m}_s \mid \alpha_0, \epsilon_0, \mathbf{u}\right)$$

- $\epsilon_1$ and $\epsilon_0$ are *discount* parameters
- When $\epsilon_1$ and $\epsilon_0$ are zero, identical to a Dirichlet distribution
- PY priors give distributions that better resemble natural language
  - Better at modeling rare words

# Pitman-Yor Predictive Distributions

- Probability of case value $c$ is now given by:

$$P(\text{case} = c \,|\, \text{tag} = s, \text{word} = w, \mathcal{D}, \alpha_1, \alpha_0, \epsilon_1, \epsilon_0) = \frac{M_{c|sw} + (\alpha_1 + \epsilon_1\, L_{\cdot|sw})\, \frac{\hat{M}_{c|s} + (\alpha_0 + \epsilon_0\, L_{\cdot|s})\, \frac{1}{C}}{\hat{N}_{\cdot|s} + \alpha_0}}{N_{\cdot|sw} + \alpha_1}$$

- Counts are now given by $M_{c|sw} = N_{c|sw} - \epsilon_1\, L_{c|sw}$ etc.

# Relationship to Bayesian *n*-gram Language Modeling

- PY priors have been used for language modeling (e.g., Teh '06)
- Kneser-Ney smoothing is equivalent to
  - Setting concentration parameters ($\alpha$s) to zero
  - Using the "minimal path" approximate inference scheme
- Kneser-Ney smoothing is one of the best smoothing methods
- Dependency models are *lexicalised* (unlike, e.g., PCFGs)
- PY priors are particularly appropriate for dependency models

# Using the Model

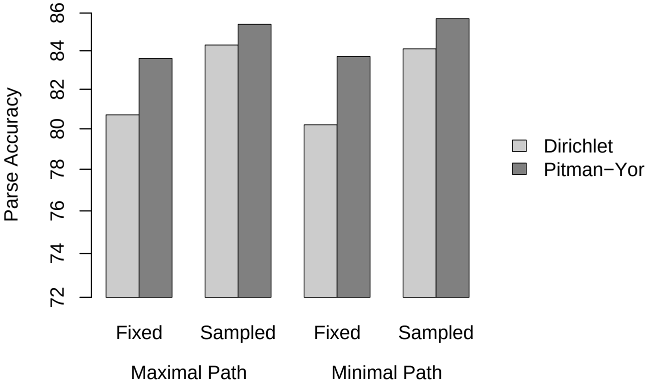
- Can now compute $P(\mathcal{D} \mid \alpha_1, \alpha_0, \epsilon_1, \epsilon_0)$
- If this were language modeling, we'd be done

- For real sentences, only words, tags and case values are known
  - Goal: infer dependency trees for real sentences
- Use training data (sentences + trees) to learn the model
- Determine trees for test sentences
  - Sample trees using Metropolis-Hastings (Johnson et al. '07)

# Parsing Experiments

- *Wall Street Journal* sections of Penn Treebank:
  - Training (sections 2–21): 39,832 sentences
  - Testing (section 23): 2,416 sentences
- Parse accuracy: percentage of parents correctly identified
- Maximum probability trees used for comparison purposes
- For efficiency, part-of-speech tags fixed to:
  - Training: "gold standard" tags from Treebank
  - Testing: tags from Ratnaparkhi's tagger ('96)

# Results: Parse Accuracy

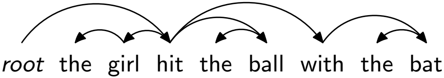



PY prior, sampled hyperparameters: 26% error reduction over Eisner

# Latent Variable Parsing Models

- Generative framework allows for inclusion of other latent variables
- e.g., "syntactic" and "semantic" topics
  - Specialized syntactic or semantic distributions over words
- Can define a (simpler) model that uses latent variables:
  - Sentences are untagged and uncased
  - Siblings are not taken into account (i.e., first order model)
  - Distribution over children depends on parent and latent state variable

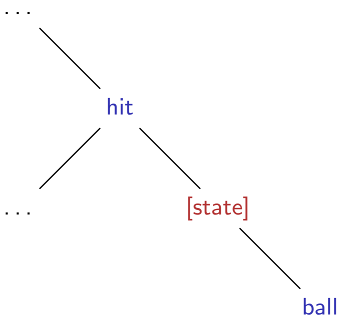
# First Order Models




- Computationally more efficient than models that consider siblings
- Children are independent of each other given their parent

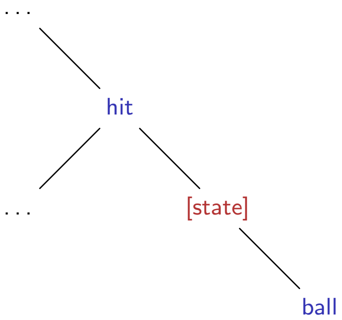
$$P(\text{girl hit . with the bat the ball}) = P(\text{the girl hit the ball with the bat})$$

# “Syntactic” Latent Variables



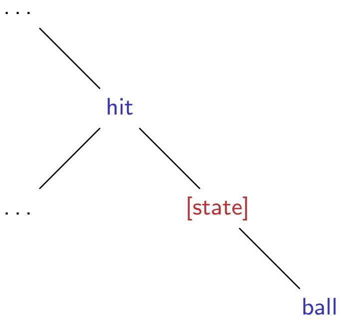
# "Syntactic" Latent Variables

- Generate a **state** given the parent word



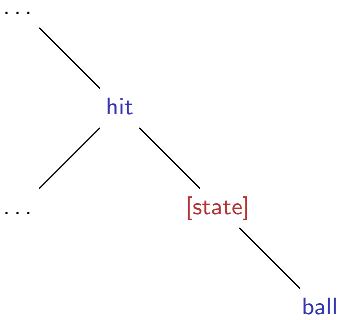
# “Syntactic” Latent Variables

- Generate a **state** given the parent word
- Generate a **word** given the just-generated state and the parent word



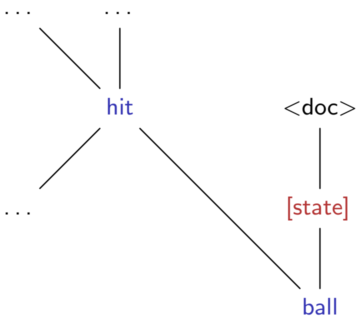
# “Syntactic” Latent Variables

- Generate a **state** given the parent word
- Generate a **word** given the just-generated state and the parent word



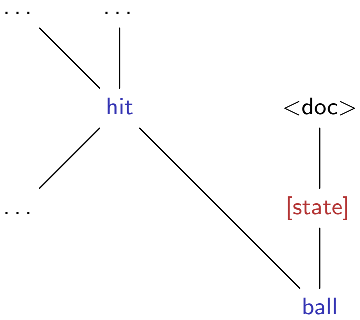

- Give all probability vectors Dirichlet priors as before

# “Semantic” Latent Variables



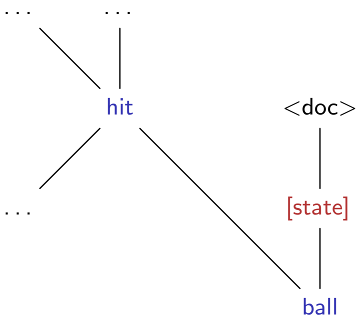
# “Semantic” Latent Variables

- Generate a **state** given the *document*, as in LDA



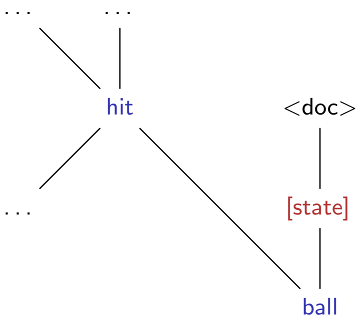
# "Semantic" Latent Variables

- Generate a **state** given the *document*, as in LDA
- Generate a **word** given the just-generated state and the parent word



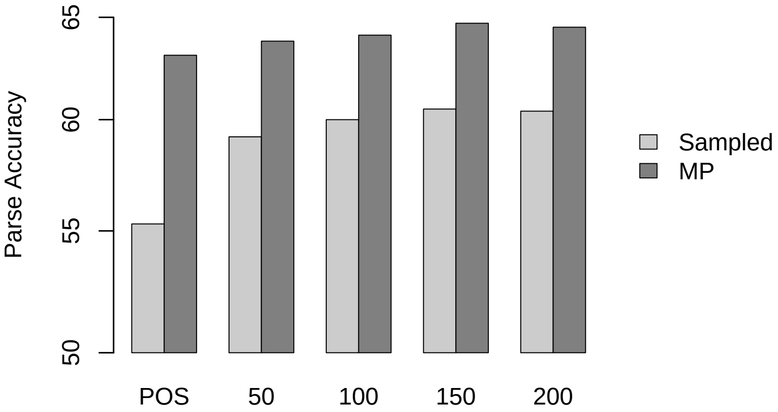
# “Semantic” Latent Variables

- Generate a **state** given the *document*, as in LDA
- Generate a **word** given the just-generated state and the parent word




- Give all probability vectors Dirichlet priors

# "Syntactic Topics": Parse Accuracy



# States Inferred from Treebank Sections 2–21

| | | | | | |
|---|---|---|---|---|---|
| president | u.s. | made | is | would | 10 |
| director | california | offered | are | will | 8 |
| officer | washington | filed | was | could | 1 |
| chairman | texas | put | has | should | 50 |
| executive | york | asked | have | can | 2 |
| head | london | approved | were | might | 15 |
| attorney | japan | announced | will | had | 20 |
| manager | canada | left | had | may | 30 |
| chief | france | held | 's | must | 25 |
| secretary | britain | bought | would | owns | 3 |

# Unsupervised Leave-One-Out Bits-Per-Word

| Model | Bits-per-Word |
|---|---|
| LDA | 9.08 |
| Deps. only | 8.75 |
| Deps. & Syntactic Topics | 8.68 |
| Deps. & Semantic Topics | 8.25 |

- The fewer the bits-per-word, the better the model

# Conclusions and Future Work

- Reinterpreted a classic dependency parser using Bayesian framework
- Parsing performance is improved by:
  - Using Pitman-Yor priors
  - Sampling hyperparameters
- Can incorporate latent variables into the model:
  - Syntactic topics that cluster parent–child relationships
  - Semantic topics, as in LDA
- Future work: syntactic + semantic topics

Questions?

wallach@cs.umass.edu
http://www.inference.phy.cam.ac.uk/hmw26/