# The Infinite Tree

**Jenny Rose Finkel, Trond Grenager, and Christopher D. Manning**
Computer Science Department, Stanford University
Stanford, CA 94305
{jrfinkel, grenager, manning}@cs.stanford.edu

## Abstract

Historically, unsupervised learning techniques have lacked a principled technique for selecting the number of unseen components. Research into non-parametric priors, such as the Dirichlet process, has enabled instead the use of *infinite models*, in which the number of hidden categories is not fixed, but can grow with the amount of training data. Here we develop the *infinite tree*, a new infinite model capable of representing recursive branching structure over an arbitrarily large set of hidden categories. Specifically, we develop three infinite tree models, each of which enforces different independence assumptions, and for each model we define a simple *direct assignment* sampling inference procedure. We demonstrate the utility of our models by doing unsupervised learning of part-of-speech tags from treebank dependency skeleton structure, achieving an accuracy of 75.34%, and by doing unsupervised splitting of part-of-speech tags, which increases the accuracy of a generative dependency parser from 85.11% to 87.35%.

## 1 Introduction

Model-based unsupervised learning techniques have historically lacked good methods for choosing the number of unseen components. For example, $k$-means or EM clustering require advance specification of the number of mixture components. But the introduction of nonparametric priors such as the *Dirichlet process* (Ferguson, 1973) enabled development of *infinite mixture models*, in which the number of hidden components is not fixed, but emerges naturally from the training data (Antoniak, 1974).

Teh et al. (2006) proposed the hierarchical Dirichlet process (HDP) as a way of applying the Dirichlet process (DP) to more complex model forms, so as to allow multiple, group-specific, infinite mixture models to *share* their mixture components. The closely related *infinite hidden Markov model* is an HMM in which the transitions are modeled using an HDP, enabling unsupervised learning of sequence models when the number of hidden states is unknown (Beal et al., 2002; Teh et al., 2006).

We extend this work by introducing the *infinite tree model*, which represents recursive branching structure over a potentially infinite set of hidden states. Such models are appropriate for the syntactic dependency structure of natural language. The hidden states represent word categories ("tags"), the observations they generate represent the words themselves, and the tree structure represents syntactic dependencies between pairs of tags.

To validate the model, we test unsupervised learning of tags conditioned on a given dependency tree structure. This is useful, because coarse-grained syntactic categories, such as those used in the Penn Treebank (PTB), make insufficient distinctions to be the basis of accurate syntactic parsing (Charniak, 1996). Hence, state-of-the-art parsers either supplement the part-of-speech (POS) tags with the lexical forms themselves (Collins, 2003; Charniak, 2000), manually split the tagset into a finer-grained one (Klein and Manning, 2003a), or learn finer grained tag distinctions using a heuristic learning procedure (Petrov et al., 2006). We demonstrate that the tags learned with our model are correlated with the PTB POS tags, and furthermore that they improve the accuracy of an automatic parser when used in training.

## 2 Finite Trees

We begin by presenting three *finite* tree models, each with different independence assumptions.

*Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 272–279,
Prague, Czech Republic, June 2007. ©2007 Association for Computational Linguistics

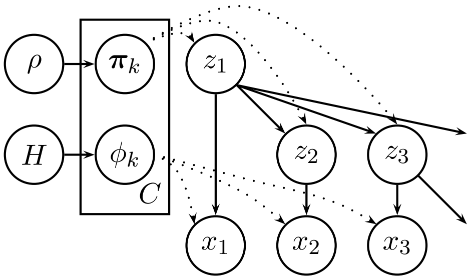


Figure 1: A graphical representation of the *finite* Bayesian tree model with independent children. The *plate* (rectangle) indicates that there is one copy of the model parameter variables for each state $k \le C$.

### 2.1 Independent Children

In the first model, children are generated independently of each other, conditioned on the parent. Let $t$ denote both the tree and its root node, $c(t)$ the list of children of $t$, $c_i(t)$ the $i^{th}$ child of $t$, and $p(t)$ the parent of $t$. Each tree $t$ has a hidden state $z_t$ (in a syntax tree, the tag) and an observation $x_t$ (the word).[1] The probability of a tree is given by the recursive definition:[2]

$$\mathrm{P}_{tr}(t) = \mathrm{P}(x_t|z_t) \prod\nolimits_{t' \in c(t)} \mathrm{P}(z_{t'}|z_t)\mathrm{P}_{tr}(t')$$

To make the model Bayesian, we must define random variables to represent each of the model's parameters, and specify prior distributions for them. Let each of the hidden state variables have $C$ possible values which we will index with $k$. Each state $k$ has a distinct distribution over observations, parameterized by $\phi_k$, which is distributed according to a prior distribution over the parameters $H$:

$$\phi_k | H \sim H$$

We generate each observation $x_t$ from some distribution $F(\phi_{z_t})$ parameterized by $\phi_{z_t}$ specific to its corresponding hidden state $z_t$. If $F(\phi_k)$s are multinomials, then a natural choice for $H$ would be a Dirichlet distribution.[3]

The hidden state $z_{t'}$ of each child is distributed according to a multinomial distribution $\pi_{z_t}$ specific to the hidden state $z_t$ of the parent:

$$x_t | z_t \sim F(\phi_{z_t})$$
$$z_{t'} | z_t \sim \text{Multinomial}(\pi_{z_t})$$

Each multinomial over children $\boldsymbol{\pi}_k$ is distributed according to a Dirichlet distribution with parameter $\rho$:

$$\boldsymbol{\pi}_k | \rho \sim \text{Dirichlet}(\rho, \ldots, \rho)$$

This model is presented graphically in Figure 1.

### 2.2 Simultaneous Children

The independent child model adopts strong independence assumptions, and we may instead want models in which the children are conditioned on more than just the parent's state. Our second model thus generates the states of all of the children $c(t)$ simultaneously:

$$\mathrm{P}_{tr}(t) = \mathrm{P}(x_t|z_t)\mathrm{P}((z_{t'})_{t' \in c(t)}|z_t) \prod\nolimits_{t' \in c(t)} \mathrm{P}_{tr}(t')$$

where $(z_{t'})_{t' \in c(t)}$ indicates the list of tags of the children of $t$. To parameterize this model, we replace the multinomial distribution $\boldsymbol{\pi_k}$ over states with a multinomial distribution $\boldsymbol{\lambda_k}$ over lists of states.[4]

### 2.3 Markov Children

The very large domain size of the child lists in the simultaneous child model may cause problems of sparse estimation. Another alternative is to use a first-order Markov process to generate children, in which each child's state is conditioned on the previous child's state:

$$\mathrm{P}_{tr}(t) = \mathrm{P}(x_t|z_t) \prod\nolimits_{i=1}^{|c(t)|} \mathrm{P}(z_{c_i(t)}|z_{c_{i-1}(t)}, z_t)\mathrm{P}_{tr}(t')$$

For this model, we augment all child lists with a distinguished *start node*, $c_0(t)$, which has as its state a distinguished *start state*, allowing us to capture the unique behavior of the first (observed) child. To parameterize this model, note that we will need to define $C(C+1)$ multinomials, one for each parent state and preceding child state (or a distinguished start state).

## 3 To Infinity, and Beyond ...

This section reviews needed background material for our approach to making our tree models infinite.

### 3.1 The Dirichlet Process

Suppose we model a document as a *bag of words* produced by a mixture model, where the mixture components might be *topics* such as business, politics, sports, etc. Using this model we can generate a

[1] To model length, every child list ends with a distinguished *stop node*, which has as its state a distinguished *stop state*.

[2] We also define a distinguished node $t_0$, which generates the root of the entire tree, and $P(x_{t_0}|z_{t_0}) = 1$.

[3] A Dirichlet *distribution* is a distribution over the possible parameters of a multinomial distributions, and is distinct from the Dirichlet *process*.

[4] This requires stipulating a maximum list length.

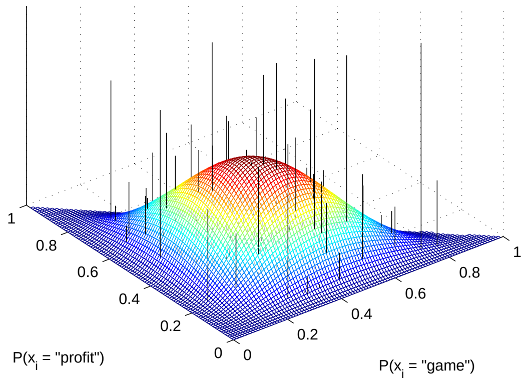
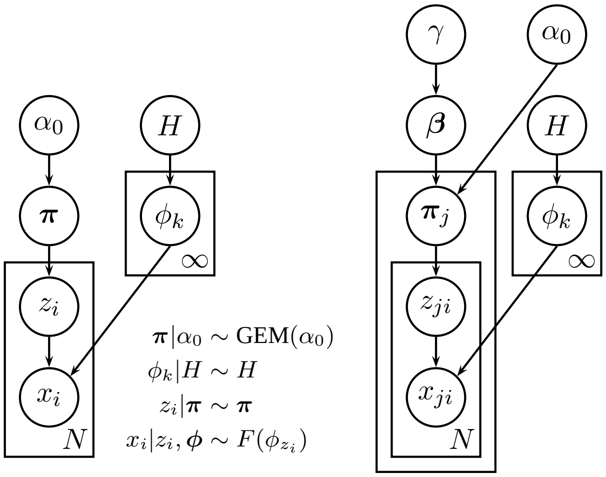


Figure 2: Plot of the density function of a Dirichlet distribution $H$ (the surface) as well as a draw $G$ (the vertical lines, or *sticks*) from a Dirichlet process DP$(\alpha_0, H)$ which has $H$ as a base measure. Both distributions are defined over a simplex in which each point corresponds to a particular multinomial distribution over three possible words: “profit”, “game”, and “election”. The placement of the sticks is drawn from the distribution $H$, and is independent of their lengths, which is drawn from a *stick-breaking* process with parameter $\alpha_0$.



(a) (b)

Figure 3: A graphical representation of a simple Dirichlet process mixture model (left) and a hierarchical Dirichlet process model (right). Note that we show the *stick-breaking* representations of the models, in which we have factored $G \sim$ DP$(\alpha_0, H)$ into two sets of variables: $\pi$ and $\phi$.

document by first generating a distribution over topics $\boldsymbol{\pi}$, and then for each position $i$ in the document, generating a topic $z_i$ from $\boldsymbol{\pi}$, and then a word $x_i$ from the topic specific distribution $\phi_{z_i}$. The word distributions $\phi_k$ for each topic $k$ are drawn from a base distribution $H$. In Section 2, we sample $C$ multinomials $\phi_k$ from $H$. In the infinite mixture model we sample an infinite number of multinomials from $H$, using the Dirichlet process.

Formally, given a base distribution $H$ and a concentration parameter $\alpha_0$ (loosely speaking, this controls the relative sizes of the topics), a Dirichlet process DP$(\alpha_0, H)$ is the distribution of a discrete random probability measure $G$ over the same (possibly continuous) space that $H$ is defined over; thus *it is a measure over measures*. In Figure 2, the sticks (vertical lines) show a draw $G$ from a Dirichlet process where the base measure $H$ is a Dirichlet distribution over 3 words. A draw comprises of an infinite number of sticks, and each corresponding topic.

We factor $G$ into two coindexed distributions: $\boldsymbol{\pi}$, a distribution over the integers, where the integer represents the index of a particular topic (i.e., the height of the sticks in the figure represent the probability of the topic indexed by that stick) and $\boldsymbol{\phi}$, representing the word distribution of each of the topics (i.e., the location of the sticks in the figure). To generate $\boldsymbol{\pi}$ we first generate an infinite sequence of variables $\boldsymbol{\pi'} = (\pi'_k)_{k=1}^{\infty}$, each of which is distributed according to the Beta distribution:

$$\pi'_k|\alpha_0 \sim \text{Beta}(1, \alpha_0)$$

Then $\boldsymbol{\pi} = (\pi_k)_{k=1}^{\infty}$ is defined as:

$$\pi_k = \pi'_k \prod_{i=1}^{k-1}(1 - \pi'_i)$$

Following Pitman (2002) we refer to this process as $\pi \sim \text{GEM}(\alpha_0)$. It should be noted that $\sum_{k=1}^{\infty} \pi_k = 1$,[5] and $P(i) = \pi_i$. Then, according to the DP, $P(\phi_i) = \pi_i$. The complete model, is shown graphically in Figure 3(a).

To build intuition, we walk through the process of generating from the infinite mixture model for the document example, where $x_i$ is the word at position $i$, and $z_i$ is its topic. $F$ is a multinomial distribution parameterized by $\boldsymbol{\phi}$, and $H$ is a Dirichlet distribution. Instead of generating all of the infinite mixture components $(\pi_k)_{k=1}^{\infty}$ at once, we can build them up incrementally. If there are $K$ known topics, we represent only the known elements $(\pi_k)_{k=1}^{K}$ and represent the remaining probability mass $\pi_u =$

[5] This is called the *stick-breaking* construction: we start with a stick of unit length, representing the entire probability mass, and successively break bits off the end of the stick, where the proportional amount broken off is represented by $\pi'_k$ and the absolute amount is represented by $\pi_k$.

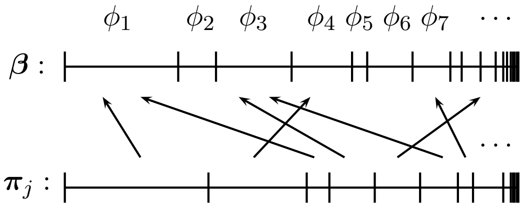


Figure 4: A graphical representation of $\boldsymbol{\pi}_j$, a broken stick, which is distributed according to a DP with a broken stick $\boldsymbol{\beta}$ as a base measure. Each $\beta_k$ corresponds to a $\phi_k$.

$1 - (\sum_{k=1}^{K} \pi_k)$. Initially we have $\pi_u = 1$ and $\boldsymbol{\phi} = ()$.

For the $i$th position in the document, we first draw a topic $z_i \sim \boldsymbol{\pi}$. If $z_i \neq u$, then we find the coindexed topic $\phi_{z_i}$. If $z_i = u$, the unseen topic, we make a draw $b \sim$ Beta$(1, \alpha_0)$ and set $\pi_{K+1} = b\pi_u$ and $\pi_u^{new} = (1 - b)\pi_u$. Then we draw a parameter $\phi_{K+1} \sim H$ for the new topic, resulting in $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_{K+1}, \pi_u^{new})$ and $\boldsymbol{\phi} = (\phi_1, \ldots, \phi_{K+1})$. A word is then drawn from this topic and emitted by the document.

### 3.2 The Hierarchical Dirichlet Process

Let's generalize our previous example to a corpus of documents. As before, we have a set of shared topics, but now each document has its *own characteristic distribution* over these topics. We represent topic distributions both locally (for each document) and globally (across all documents) by use of a hierarchical Dirichlet process (HDP), which has a local DP for each document, in which *the base measure is itself a draw from another, global, DP.*

The complete HDP model is represented graphically in Figure 3(b). Like the DP, it has global broken stick $\boldsymbol{\beta} = (\beta_k)_{k=1}^{\infty}$ and topic specific word distribution parameters $\boldsymbol{\phi} = (\phi_k)_{k=1}^{\infty}$, which are coindexed. It differs from the DP in that it also has local broken sticks $\boldsymbol{\pi}_j$ for each group $j$ (in our case documents). While the global stick $\boldsymbol{\beta} \sim$ GEM$(\gamma)$ is generated as before, the local sticks $\boldsymbol{\pi}_j$ are distributed according to a DP with base measure $\boldsymbol{\beta}$: $\boldsymbol{\pi}_j \sim$ DP$(\alpha_0, \boldsymbol{\beta})$.

We illustrate this generation process in Figure 4. The upper unit line represents $\boldsymbol{\beta}$, where the size of segment $k$ represents the value of element $\beta_k$, and the lower unit line represents $\boldsymbol{\pi}_j \sim$ DP$(\alpha_0, \boldsymbol{\beta})$ for a particular group $j$. Each element of the lower stick was sampled from a particular element of the upper stick, and elements of the upper stick may be sampled multiple times or not at all; on average, larger elements will be sampled more often. Each element $\boldsymbol{\beta}_k$, as well as all elements of $\boldsymbol{\pi}_j$ that were sampled from it, corresponds to a particular $\phi_k$. Critically, several distinct $\boldsymbol{\pi}_j$ can be sampled from the same $\beta_k$ and hence share $\phi_k$; *this is how components are shared among groups.*

For concreteness, we show how to generate a corpus of documents from the HDP, generating one document at a time, and incrementally constructing our infinite objects. Initially we have $\beta_u = 1$, $\boldsymbol{\phi} = ()$, and $\pi_{ju} = 1$ for all $j$. We start with the first position of the first document and draw a local topic $y_{11} \sim \boldsymbol{\pi}_1$, which will return $u$ with probability 1. Because $y_{11} = u$ we must make a draw from the base measure, $\boldsymbol{\beta}$, which, because this is the first document, will also return $u$ with probability 1. We must now break $\beta_u$ into $\beta_1$ and $\beta_u^{new}$, and break $\pi_{1u}$ into $\pi_{11}$ and $\pi_{1u}^{new}$ in the same manner presented for the DP. Since $\pi_{11}$ now corresponds to global topic 1, we sample the word $x_{11} \sim \text{Multinomial}(\phi_1)$. To sample each subsequent word $i$, we first sample the local topic $y_{1i} \sim \boldsymbol{\pi}_1$. If $y_{1i} \neq u$, and $\pi_{1y_{1i}}$ corresponds to $\beta_k$ in the global stick, then we sample the word $x_{1i} \sim \text{Multinomial}(\phi_k)$. Once the first document has been sampled, subsequent documents are sampled in a similar manner; initially $\pi_{ju} = 1$ for document $j$, while $\boldsymbol{\beta}$ continues to grow as more documents are sampled.

## 4 Infinite Trees

We now use the techniques from Section 3 to create infinite versions of each tree model from Section 2.

### 4.1 Independent Children

The changes required to make the Bayesian independent children model infinite don't affect its basic structure, as can be witnessed by comparing the graphical depiction of the infinite model in Figure 5 with that of the finite model in Figure 1. The instance variables $z_t$ and $x_t$ are parameterized as before. The primary change is that the number of copies of the state plate is infinite, as are the number of variables $\boldsymbol{\pi}_k$ and $\phi_k$.

Note also that each distribution over possible child states $\boldsymbol{\pi}_k$ must also be infinite, since the number of possible child states is potentially infinite. We achieve this by representing each of the $\boldsymbol{\pi}_k$ variables as a broken stick, and adopt the same approach of

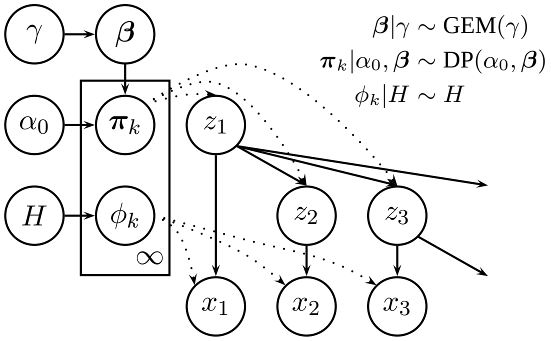

$\boldsymbol{\beta}|\gamma \sim \mathrm{GEM}(\gamma)$
$\boldsymbol{\pi}_k|\alpha_0,\boldsymbol{\beta} \sim \mathrm{DP}(\alpha_0,\boldsymbol{\beta})$
$\phi_k|H \sim H$
$\gamma$
$\boldsymbol{\beta}$
$\alpha_0$
$\boldsymbol{\pi}_k$
$H$
$\phi_k$
$\infty$
$z_1$
$z_2$
$z_3$
$x_1$
$x_2$
$x_3$

Figure 5: A graphical representation of the *infinite* independent child model.

sampling each $\boldsymbol{\pi}_k$ from a DP with base measure $\boldsymbol{\beta}$. For the dependency tree application, $\phi_k$ is a vector representing the parameters of a multinomial over words, and $H$ is a Dirichlet distribution.

The infinite hidden Markov model (iHMM) or HDP-HMM (Beal et al., 2002; Teh et al., 2006) is a model of sequence data with transitions modeled by an HDP.[6] The iHMM can be viewed as a special case of this model, where each state (except the stop state) produces exactly one child.

### 4.2 Simultaneous Children

The key problem in the definition of the simultaneous children model is that of defining a distribution over the lists of children produced by each state, since each child in the list has as its domain the positive integers, representing the infinite set of possible states. Our solution is to construct a distribution $L_k$ over lists of states from the distribution over individual states $\boldsymbol{\pi}_k$. The obvious approach is to sample the states at each position i.i.d.:

$$\mathrm{P}((z_{t'})_{t'\in c(t)}|\boldsymbol{\pi}) = \prod_{t'\in c(t)} \mathrm{P}(z_{t'}|\boldsymbol{\pi}) = \prod_{t'\in c(t)} \pi_{z_{t'}}$$

However, we want our model to be able to represent the fact that some child lists, $\boldsymbol{c}_t$, are more or less probable than the product of the individual child probabilities would indicate. To address this, we can sample a state-conditional distribution over child lists $\lambda_k$ from a DP with $L_k$ as a base measure. Thus, we augment the basic model given in the previous section with the variables $\zeta$, $L_k$, and $\lambda_k$:

$$\begin{aligned} L_k|\boldsymbol{\pi}_k &\sim \text{Deterministic, as described above} \\ \lambda_k|\zeta, L_k &\sim \mathrm{DP}(\zeta, L_k) \\ \boldsymbol{c}_t|\lambda_k &\sim \lambda_k \end{aligned}$$

An important consequence of defining $L_k$ locally (instead of globally, using $\boldsymbol{\beta}$ instead of the $\boldsymbol{\pi}_k$s) is that the model captures not only what sequences of children a state prefers, but also the individual children that state prefers; if a state gives high probability to some particular sequence of children, then it is likely to also give high probability to other sequences containing those same states, or a subset thereof.

### 4.3 Markov Children

In the Markov children model, more copies of the variable $\boldsymbol{\pi}$ are needed, because each child state must be conditioned both on the parent state and on the state of the preceding child. We use a new set of variables $\boldsymbol{\pi}_{ki}$, where $\boldsymbol{\pi}$ is determined by the parent state $k$ and the state of the preceding sibling $i$. Each of the $\boldsymbol{\pi}_{ki}$ is distributed as $\boldsymbol{\pi}_k$ was in the basic model: $\boldsymbol{\pi}_{ki} \sim \mathrm{DP}(\alpha_0, \boldsymbol{\beta})$.

## 5 Inference

Our goal in inference is to draw a sample from the posterior over assignments of states to observations. We present an inference procedure for the infinite tree that is based on Gibbs sampling in the *direct assignment* representation, so named because we directly assign global state indices to observations.[7]

Before we present the procedure, we define a few count variables. Recall from Figure 4 that each state $k$ has a local stick $\boldsymbol{\pi}_k$, each element of which corresponds to an element of $\boldsymbol{\beta}$. In our sampling procedure, we only keep elements of $\boldsymbol{\pi}_k$ and $\boldsymbol{\beta}$ which correspond to states observed in the data. We define the variable $m_{jk}$ to be the number of elements of the finite observed portion of $\boldsymbol{\pi}_k$ which correspond to $\beta_j$ and $n_{jk}$ to be the number of observations with state $k$ whose parent's state is $j$.

We also need a few model-specific counts. For the simultaneous children model we need $\overline{n}_{j\boldsymbol{z}}$, which is

[6] The original iHMM paper (Beal et al., 2002) predates, and was the motivation for, the work presented in Teh et al. (2006), and is the origin of the term *hierarchical Dirichlet process*. However, they used the term to mean something slightly different than the HDP presented in Teh et al. (2006), and presented a sampling scheme for inference that was a heuristic approximation of a Gibbs sampler.

[7] We adapt one of the sampling schemes mentioned by Teh et al. (2006) for use in the iHMM. This paper suggests two sampling schemes for inference, but does not explicitly present them. Upon discussion with one of the authors (Y. W. Teh, 2006, p.c.), it became clear that inference using the augmented representation is much more complicated than initially thought.

the number of times the state sequence $\boldsymbol{z}$ occurred as the children of state $j$. For the Markov children model we need the count variable $\hat{n}_{jik}$ which is the number of observations for a node with state $k$ whose parent's state is $j$ and whose previous sibling's state is $i$. In all cases we represent marginal counts using dot-notation, e.g., $n_{\cdot k}$ is the total number of nodes with state $k$, regardless of parent.

Our procedure alternates between three distinct sampling stages: (1) sampling the state assignments $z$, (2) sampling the counts $m_{jk}$, and (3) sampling the global stick $\boldsymbol{\beta}$. The only modification of the procedure that is required for the different tree models is the method for computing the probability of the child state sequence given the parent state $\mathrm{P}((z_{t'})_{t'\in c(t)}|z_t)$, defined separately for each model.

**Sampling $z$.** In this stage we sample a state for each tree node. The probability of node $t$ being assigned state $k$ is given by:

$$\mathrm{P}(z_t = k|\boldsymbol{z}^{-t}, \boldsymbol{\beta}) \propto \mathrm{P}(z_t = k, (z_{t'})_{t'\in s(t)}|z_{p(t)}) \cdot \mathrm{P}((z_{t'})_{t'\in c(t)}|z_t = k) \cdot f_k^{-x_t}(x_t)$$

where $s(t)$ denotes the set of siblings of $t$, $f_k^{-x_t}(x_t)$ denotes the posterior probability of observation $x_t$ given all other observations assigned to state $k$, and $\boldsymbol{z}^{-t}$ denotes all state assignments except $z_t$. In other words, the probability is proportional to the product of three terms: the probability of the states of $t$ and its siblings given its parent $z_{p(t)}$, the probability of the states of the children $c(t)$ given $z_t$, and the posterior probability of observation $x_t$ given $z_t$. Note that if we sample $z_t$ to be a previously unseen state, we will need to extend $\boldsymbol{\beta}$ as discussed in Section 3.2.

Now we give the equations for $\mathrm{P}((z_{t'})_{t'\in c(t)}|z_t)$ for each of the models. In the independent child model the probability of generating each child is:

$$\mathrm{P}_{\text{ind}}(z_{c_i(t)} = k|z_t = j) = \frac{n_{jk} + \alpha_0\beta_k}{n_{j\cdot} + \alpha_0}$$

$$\mathrm{P}_{\text{ind}}((z_{t'})_{t'\in c(t)}|z_t = j) = \prod\nolimits_{t'\in c(t)} \mathrm{P}_{\text{ind}}(z_{t'}|z_t = j)$$

For the simultaneous child model, the probability of generating a sequence of children, $\mathbf{z}$, takes into account how many times that sequence has been generated, along with the likelihood of regenerating it:

$$\mathrm{P}_{\text{sim}}((z_{t'})_{t'\in c(t)} = \mathbf{z}|z_t = j) = \frac{\overline{n}_{j\mathbf{z}} + \zeta \mathrm{P}_{\text{ind}}(\mathbf{z}|z_t = j)}{\overline{n}_{j\cdot} + \zeta}$$

Recall that $\zeta$ denotes the concentration parameter for the sequence generating DP. Lastly, we have the Markov child model:

$$\mathrm{P}_{\text{m}}(z_{c_i(t)} = k|z_{c_{i-1}(t)} = i, z_t = j) = \frac{\hat{n}_{jik} + \alpha_0\beta_k}{\hat{n}_{ji\cdot} + \alpha_0}$$

$$\mathrm{P}_{\text{m}}((z_{t'})_{t'\in c(t)}|z_t) = \prod\nolimits_{i=1}^{|c(t)|} \mathrm{P}_{\text{m}}(z_{c_i(t)}|z_{c_{i-1}(t)}, z_t)$$

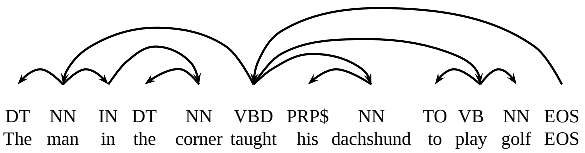


Figure 6: An example of a syntactic dependency tree where the dependencies are between tags (hidden states), and each tag generates a word (observation).

Finally, we give the posterior probability of an observation, given that $F(\phi_k)$ is Multinomial$(\phi_k)$, and that $H$ is Dirichlet$(\rho, \ldots, \rho)$. Let $N$ be the vocabulary size and $\dot{n}_k$ be the number of observations $x$ with state $k$. Then:

$$f_k^{-x_t}(x_t) = \frac{\dot{n}_{x_t k} + \rho}{\dot{n}_{\cdot k} + N\rho}$$

**Sampling $m$.** We use the following procedure, which slightly modifies one from (Y. W. Teh, 2006, p.c.), to sample each $m_{jk}$:

```
SAMPLEM$(j, k)$
if $n_{jk} = 0$
   then $m_{jk} = 0$
   else $m_{jk} = 1$
        for $i \leftarrow 2$ to $n_{jk}$
            do if rand() $< \frac{\alpha_0}{\alpha_0+i-1}$
                  then $m_{jk} = m_{jk} + 1$
return $m_{jk}$
```

**Sampling $\beta$.** Lastly, we sample $\boldsymbol{\beta}$ using the Dirichlet distribution:

$$(\beta_1, \ldots, \beta_K, \beta_u) \sim \text{Dirichlet}(m_{\cdot 1}, \ldots, m_{\cdot K}, \alpha_0)$$

## 6 Experiments

We demonstrate infinite tree models on two distinct syntax learning tasks: unsupervised POS learning conditioned on untagged dependency trees and learning a split of an existing tagset, which improves the accuracy of an automatic syntactic parser.

For both tasks, we use a simple modification of the basic model structure, to allow the trees to generate dependents on the left and the right with different distributions – as is useful in modeling natural language. The modification of the independent child tree is trivial: we have two copies of each of the variables $\boldsymbol{\pi}_k$, one each for the left and the right. Generation of dependents on the right is completely independent of that for the left. The modifications of the other models are similar, but now there are separate *sets* of $\pi_k$ variables for the Markov child model, and separate $L_k$ and $\lambda_k$ variables for the simultaneous child model, for each of the left and right.

For both experiments, we used dependency trees extracted from the Penn Treebank (Marcus et al., 1993) using the head rules and dependency extractor from Yamada and Matsumoto (2003). As is standard, we used WSJ sections 2–21 for training, section 22 for development, and section 23 for testing.

### 6.1 Unsupervised POS Learning

In the first experiment, we do unsupervised part-of-speech learning conditioned on dependency trees. To be clear, the input to our algorithm is the dependency structure skeleton of the corpus, but not the POS tags, and the output is a labeling of each of the words in the tree for word class. Since the model knows nothing about the POS annotation, the new classes have arbitrary integer names, and are not guaranteed to correlate with the POS tag definitions. We found that the choice of $\alpha_0$ and $\beta$ (the concentration parameters) did not affect the output much, while the value of $\rho$ (the parameter for the base Dirichlet distribution) made a much larger difference. For all reported experiments, we set $\alpha_0 = \beta = 10$ and varied $\rho$.

We use several metrics to evaluate the word classes. First, we use the standard approach of greedily assigning each of the learned classes to the POS tag with which it has the greatest overlap, and then computing tagging accuracy (Smith and Eisner, 2005; Haghighi and Klein, 2006).[8] Additionally, we compute the mutual information of the learned clusters with the gold tags, and we compute the cluster F-score (Ghosh, 2003). See Table 1 for results of the different models, parameter settings, and metrics. Given the variance in the number of classes learned it is a little difficult to interpret these results, but it is clear that the Markov child model is the best; it achieves superior performance to the independent child model on all metrics, while learning fewer word classes. The poor performance of the simultaneous model warrants further investigation, but we observed that the distributions learned by that model are far more spiked, potentially due to double counting of tags, since the sequence probabilities are already based on the local probabilities.

| **Model** | $\rho$ | **# Classes** | **Acc.** | **MI** | **F1** |
|---|---|---|---|---|---|
| Indep. | 0.01 | 943 | 67.89 | 2.00 | 48.29 |
| | 0.001 | 1744 | 73.61 | 2.23 | 40.80 |
| | 0.0001 | 2437 | 74.64 | 2.27 | 39.47 |
| Simul. | 0.01 | 183 | 21.36 | 0.31 | 21.57 |
| | 0.001 | 430 | 15.77 | 0.09 | 13.80 |
| | 0.0001 | 549 | 16.68 | 0.12 | 14.29 |
| Markov | 0.01 | 613 | 68.53 | 2.12 | 49.82 |
| | 0.001 | 894 | 75.34 | 2.31 | 48.73 |

Table 1: Results of part unsupervised POS tagging on the different models, using a greedy accuracy measure.

For comparison, Haghighi and Klein (2006) report an unsupervised baseline of 41.3%, and a best result of 80.5% from using hand-labeled prototypes and distributional similarity. However, they train on less data, and learn fewer word classes.

### 6.2 Unsupervised POS Splitting

In the second experiment we use the infinite tree models to learn a refinement of the PTB tags. We initialize the set of hidden states to the set of PTB tags, and then, during inference, constrain the sampling distribution over hidden state $z_t$ at each node $t$ to include only states that are a refinement of the annotated PTB tag at that position. The output of this training procedure is a new annotation of the words in the PTB with the learned tags. We then compare the performance of a generative dependency parser trained on the new refined tags with one trained on the base PTB tag set. We use the generative dependency parser distributed with the Stanford factored parser (Klein and Manning, 2003b) for the comparison, since it performs simultaneous tagging and parsing during testing. In this experiment, unlabeled, directed, dependency parsing accuracy for the best model increased from 85.11% to 87.35%, a 15% error reduction. See Table 2 for the full results over all models and parameter settings.

## 7 Related Work

The HDP-PCFG (Liang et al., 2007), developed at the same time as this work, aims to learn state splits for a binary-branching PCFG. It is similar to our simultaneous child model, but with several important distinctions. As discussed in Section 4.2, in our model each state has a DP over sequences, with a base distribution that is defined over the local child

[8] The advantage of this metric is that it's comprehensible. The disadvantage is that it's easy to inflate by adding classes.

| Model | $\rho$ | Accuracy |
|---|---|---|
| Baseline | – | 85.11 |
| Independent | 0.01 | 86.18 |
| | 0.001 | 85.88 |
| Markov | 0.01 | 87.15 |
| | 0.001 | 87.35 |

Table 2: Results of untyped, directed dependency parsing, where the POS tags in the training data have been split according to the various models. At test time, the POS tagging and parsing are done simultaneously by the parser.

state probabilities. In contrast, Liang et al. (2007) define a global DP over sequences, with the base measure defined over the global state probabilities, $\boldsymbol{\beta}$; locally, each state has an HDP, with this global DP as the base measure. We believe our choice to be more linguistically sensible: in our model, for a particular state, dependent sequences which are similar to one another increase one another's likelihood. Additionally, their modeling decision made it difficult to define a Gibbs sampler, and instead they use variational inference. Earlier, Johnson et al. (2007) presented *adaptor grammars*, which is a very similar model to the HDP-PCFG. However they did not confine themselves to a binary branching structure and presented a more general framework for defining the process for splitting the states.

## 8 Discussion and Future Work

We have presented a set of novel infinite tree models and associated inference algorithms, which are suitable for representing syntactic dependency structure. Because the models represent a potentially infinite number of hidden states, they permit unsupervised learning algorithms which naturally select a number of word classes, or tags, based on qualities of the data. Although they require substantial technical background to develop, the learning algorithms based on the models are actually simple in form, requiring only the maintenance of counts, and the construction of sampling distributions based on these counts. Our experimental results are preliminary but promising: they demonstrate that the model is capable of capturing important syntactic structure.

Much remains to be done in applying infinite models to language structure, and an interesting extension would be to develop inference algorithms that permit completely unsupervised learning of dependency structure.

## Acknowledgments

Many thanks to Yeh Whye Teh for several enlightening conversations, and to the following members (and honorary member) of the Stanford NLP group for comments on an earlier draft: Thad Hughes, David Hall, Surabhi Gupta, Ani Nenkova, Sebastian Riedel. This work was supported by a Scottish Enterprise Edinburgh-Stanford Link grant (R37588), as part of the EASIE project, and by the Advanced Research and Development Activity (ARDA)'s Advanced Question Answering for Intelligence (AQUAINT) Phase II Program.

## References

C. E. Antoniak. 1974. Mixtures of Dirichlet processes with applications to Bayesian nonparametrics. *Annals of Statistics*, 2:1152–1174.

M.J. Beal, Z. Ghahramani, and C.E. Rasmussen. 2002. The infinite hidden Markov model. In *Advances in Neural Information Processing Systems*, pages 577–584.

E. Charniak. 1996. Tree-bank grammars. In *AAAI 1996*, pages 1031–1036.

E. Charniak. 2000. A maximum-entropy-inspired parser. In *HLT-NAACL 2000*, pages 132–139.

M. Collins. 2003. Head-driven statistical models for natural language parsing. *Computational Linguistics*, 29(4):589–637.

T. S. Ferguson. 1973. A Bayesian analysis of some nonparametric problems. *Annals of Statistics*, 1:209–230.

J. Ghosh. 2003. Scalable clustering methods for data mining. In N. Ye, editor, *Handbook of Data Mining*, chapter 10, pages 247–277. Lawrence Erlbaum Assoc.

A. Haghighi and D. Klein. 2006. Prototype-driven learning for sequence models. In *HLT-NAACL 2006*.

M. Johnson, T. Griffiths, and S. Goldwater. 2007. Adaptor grammars: A framework for specifying compositional nonparametric Bayesian models. In *NIPS 2007*.

D. Klein and C. D. Manning. 2003a. Accurate unlexicalized parsing. In *ACL 2003*.

D. Klein and C. D. Manning. 2003b. Factored A* search for models over sequences and trees. In *IJCAI 2003*.

P. Liang, S. Petrov, D. Klein, and M. Jordan. 2007. Nonparametric PCFGs using Dirichlet processes. In *EMNLP 2007*.

M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

S. Petrov, L. Barrett, R. Thibaux, and D. Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *ACL 44/COLING 21*, pages 433–440.

J. Pitman. 2002. Poisson-Dirichlet and GEM invariant distributions for split-and-merge transformations of an interval partition. *Combinatorics, Probability and Computing*, 11:501–514.

N. A. Smith and J. Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *ACL 2005*.

Y. W. Teh, M.I. Jordan, M. J. Beal, and D.M. Blei. 2006. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101:1566–1581.

H. Yamada and Y. Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of IWPT*, pages 195–206.