An Overview of Nonparametric Bayesian Models and Applications to Natural Language Processing

Narges Sharif-Razavian and **Andreas Zollmann**
School of Computer Science
Carnegie Mellon University
Pittsburgh, USA
{nsharifr,zollmann}@cs.cmu.edu

Abstract

This paper provides an overview of nonparametric Bayesian models relevant to natural language processing (NLP) tasks. We first introduce Bayesian parametric methods, followed by nonparametric Bayesian modeling based on the most common nonparametric prior, the Dirichlet Process. We give characterizations of the Dirichlet Process via the Polya urn scheme, the related Chinese restaurant metaphor, and the stick-breaking construction. We will also introduce two generalizations of Dirichlet Processes: Hierarchical Dirichlet Processes and Pitman-Yor Processes. We also review four recently proposed nonparametric Bayesian solutions to the NLP tasks of word segmentation, phrase extraction and alignment, context free parsing, and language modeling.

1 Introduction

Probabilistic models now play a major role in nearly every natural language processing task. These models define a probability distribution over the output space, where the output can be any structural or analytical representation of the input, such as part of speech tags, parse trees or even the translation of the input in another language. In general, these models have adjustable parameters that determine the distribution.

Essentially there are two possible ways to define the parameters of these models. Maximum-likelihood estimation, which finds the value of the parameter that maximizes the likelihood of the current data; and Bayesian modeling, which assigns a probability distribution on the parameters themselves.

The number of the parameters of the model can be assumed to be fixed, or it can be assumed to be variable and possibly infinite. The term *nonparametric* refers to the latter set of models. Not committing to a fixed number of parameters can have the advantages of making the model conceptually simpler, as well as more adaptable to the size of the training corpus.

There can be Bayesian and non-Bayesian nonparametric models. K-Nearest-Neighbor (KNN) models are examples of non-Bayesian nonparametric probabilistic models. The parameters that the KNN models can not fix in advance are the size of the window into which the K nearest neighbors will fall, for each data point in the input.

There can also be Bayesian nonparametric models. These models use priors with an infinite number of parameters to model the data. Being Bayesian allows the model to avoid having to choose a single value for each parameter, and being nonparametric allows the model to increase the dimensionality of the parameters as more data becomes available.

In this paper, we provide an overview of nonparametric Bayesian models relevant to natural language processing tasks. Section 2 starts with a background on Bayesian parametric models, the Dirichlet distribution, and Bayesian parametric mixture modeling. We then extend our formulation of parametric Bayesian modeling to nonparametric models, introducing the most common nonparametric prior, the Dirichlet Process, and a generalization, the Hierarchical Dirichlet Process, which allows the model's components to have dependency relations between themselves. We will also introduce the Pitman-Yor Process, another generalization of Dirichlet Processes. In Section 3, we will review the results of four recent applications based on models introduced in Section 2: word segmentation, phrase extraction and alignment, context free parsing, and language modeling. We conclude in Section 4, giving some possible future directions and references for further reading.

2 Bayesian Methods

2.1 Background

Given the task of learning to predict a label y for an observed variable x based on an iid sample of training instances $x_1, \dots, x_N$ and usually their respective labels $y_1, \dots, y_N$, the statistical estimation approach is to devise a class of joint probability distributions—also called a *model*—$p_\theta(x, y)$ over x and y, parameterized by θ. Predicting the label then becomes the task of finding

$$\arg\max_y p_\theta(y|x) = \arg\max_y p_\theta(x, y) \ .$$

A common method for determining the parameters θ is *maximum-likelihood estimation (MLE)*, chosing the θ that maximizes the probability of the training corpus $D = (x_1, y_1), \dots, (x_N, y_N)$:

$$\begin{aligned}\hat{\theta} &= \arg\max_\theta p_\theta(D) \\ &= \arg\max_\theta p_\theta(x_1, y_1) \cdots p_\theta(x_N, y_N)\end{aligned}$$

or, if the labels are not given,

$$\begin{aligned}\hat{\theta} &= \arg\max_\theta p_\theta(x_1) \cdots p_\theta(x_N) \\ &= \arg\max_\theta \prod_{i=1}^{N} \sum_{y \in \mathcal{Y}} p_\theta(x_i, y)\end{aligned}$$

where $\mathcal{Y}$ is the set of possible labels.

Maximum-likelihood estimation is prone to overfitting: The estimated distribution only acknowledges events that occurred in the training data. As a result, new, previously unseen, events occurring during testing are assigned probability zero. An elegant way to avoid overfitting is treating the model parameters as random variables themselves, thus assigning *prior distributions* to them. Instead of choosing θ to maximize the likelihood of the data, we now work with a *posterior distribution*

$$p(\theta|D) = \frac{p(D|\theta)p(\theta)}{p(D)} \propto p(D|\theta)p(\theta) \ .$$

2.2 The coin flip example

As a simple example, consider the case of a weighted coin: x is a constant, and y is the outcome of a coin flip, 'head' or 'tail'. Our model is the class of Bernoulli distributions indexed by θ:

$$p_\theta(x, y) = \begin{cases} \theta & \text{if } y = \text{'head'} \\ 1 - \theta & \text{else} \end{cases}$$

Therefore, the likelihood function $p_\theta(D)$ is Binomial:

$$p_\theta(D) = \theta^{\#_H}(1-\theta)^{\#_T}$$

where $\#_H$ is the number of heads encountered in the training data D, and $\#_T$ the number of tails, and it is easy to show that maximum-likelihood estimation will estimate θ as the relative frequency of heads in the training corpus:

$$\hat{\theta} = \frac{\#_H}{\#_H + \#_T}$$

Imagine now what happens to our estimator when we toss a typical sequence such as 'head', 'tail', 'tail' and repeatedly estimate based on the training data obtained so far: After the first toss, the MLE (and thus the estimated probability of the coin's tossing 'head') is $\hat{\theta} = 1$, i.e., we are certain that the coin only tosses head. Then the MLE changes to $\hat{\theta} = 0.5$, a fair coin. After the third toss, the MLE decreases to $1/3$, now estimating the coin as strongly biased towards tails.

To counter this problem of overfitting on small training samples, we can equip θ with a prior. It is convenient to chose the prior in such a way that the resulting posterior distribution is of the same family as the prior and thus has a closed-form solution. Such priors are called *conjugate* to the model distribution. The conjugate prior to the Binomial is the Beta distribution $\mathrm{Beta}(\alpha_H, \alpha_T)$, given (up to proportionality) by the pdf:

$$p(\theta; \alpha_H, \alpha_T) \propto \theta^{(\alpha_H - 1)}(1-\theta)^{(\alpha_T - 1)}$$

The resulting posterior is thus:

$$\begin{aligned}p(\theta|D) &\propto p(D|\theta)p(\theta)\\ &\propto \theta^{\#_H}(1-\theta)^{\#_T}\theta^{(\alpha_H-1)}(1-\theta)^{(\alpha_T-1)}\\ &= \theta^{(\#_H+\alpha_H-1)}(1-\theta)^{(\#_T+\alpha_T-1)}\end{aligned}$$

and therefore we have:

$$\theta|D \sim \mathrm{Beta}(\#_H+\alpha_H, \#_T+\alpha_T)$$

To estimate e.g. the probability of a 'head' toss given our sequence of 'training' tosses in this Bayesian setting, we compute that probability directly, marginalizing over the prior θ:

$$\begin{aligned}P(\text{'head'}|D) &= \int_\theta P(\text{'head'}|\theta)\, p(\theta|D)\, d\theta\\ &= \int_\theta \theta\, p(\theta|D)\, d\theta\end{aligned}$$

As observed above, $p(\theta|D)$ is the pdf of the $\mathrm{Beta}(\#_H+\alpha_H, \#_T+\alpha_T)$ distribution. Therefore, our 'head' toss probability is the mean of that Beta distribution, which is given by

$$\frac{\#_H+\alpha_H}{\#_H+\alpha_H+\#_T+\alpha_T}$$

Note how this term is dominated by the prior parameters α_H and α_T when the training sample D is small, and how it converges to the maximum-likelihood estimate $\hat{\theta} = \#_H/(\#_H+\#_T)$ as D becomes large. A maximum-likelihood way of interpreting the Bayesian model for this simple example is imagining the training data to be prepended by some pseudo coin flips: α_H head tosses and α_T tail tosses.

2.3 The Dirichlet Distribution

Let us now generalize the coin flip example to the case where y can have K different outcomes, which we will simply call $1, 2, \ldots, K$. Each toss is thus distributed according to a single-trial $\mathrm{Multinomial}(\beta = \langle\beta_1, \ldots, \beta_K\rangle)$ (where β is non-negative and $\sum_i \beta_i = 1$) with pmf:

$$p(k;\beta) = \beta_k$$

Now let the parameter vector β itself be drawn from a $\mathrm{Dirichlet}(\alpha_1, \ldots, \alpha_K)$ distribution (where all $\alpha_i > 0$), the multivariate generalization of the Beta distribution. It has the following pdf, defined for all non-negative β with $\sum_i \beta_i = 1$:

$$p(\beta;\alpha_1,\ldots,\alpha_K) \propto \prod_{k=1}^{K} \beta_k^{(\alpha_k-1)}$$

As the reader might have guessed already, this distribution is conjugate to the $\mathrm{Multinomial}$: Given n iid tosses $y_1, \ldots, y_n$ according to $\mathrm{Multinomial}(\beta)$, the posterior distribution of β, and thus the distribution of the distribution of a new toss y_{n+1} given the previous tosses, is as follows (where $n_k = \sum_{i=1}^n I(y_i = k)$ is the total number of outcomes k amongst tosses $y_1, \ldots, y_n$):

$$\begin{aligned}p(\beta|y_1,\ldots,y_n) &\propto p(y_1,\ldots,y_n|\beta)p(\beta)\\ &\propto \prod_{k=1}^{K}\beta_k^{n_k} \times \prod_{k=1}^{K}\beta_k^{(\alpha_k-1)}\\ &= \prod_{k=1}^{K}\beta_k^{(n_k+\alpha_k-1)}\end{aligned}$$

Therefore, $\beta|y_1, \ldots, y_n$ has a

$$\mathrm{Dirichlet}(n_1+\alpha_1, \ldots, n_K+\alpha_K)$$

distribution. Marginalizing over β gives us the expected probability of a new toss to have a certain outcome:

$$\begin{aligned}p(y_{n+1}=k) &= E\left[\beta_k|\, y_1,\ldots,y_n\right]\\ &= \frac{n_k+\alpha_k}{n+\sum_{j=1}^{K}\alpha_j}\end{aligned} \quad (1)$$

where the last equality results from the fact that the mean of $\mathrm{Dirichlet}(\gamma_1, \ldots, \gamma_K)$ is given by:

$$\left(\frac{\gamma_1}{\sum_j \gamma_j}, \ldots, \frac{\gamma_K}{\sum_j \gamma_j}\right)$$

To ease the transition to the Dirichlet Process, introduced later in this paper, it is convenient to think of our $\mathrm{Multinomial}$ parameter β not as a vector in $[0,1]^K$, but rather as a mapping $\beta : \{1, \ldots, K\} \to [0,1]$. This way, β can directly be interpreted as a probability distribution over the event space $\{1, \ldots, K\}$, and we can express the two-step generative story without the $\mathrm{Multinomial}$ notation:

$$y \sim \beta$$

$$\beta \sim \mathrm{Dirichlet}(\alpha_1, \ldots, \alpha_K)$$

2.4 Bayesian finite mixture models

To pave the way for typical applications of the nonparametric Dirichlet Process and Hierarchical Dirichlet Process models, we follow (Liang et al., 2007) and consider the example of a Bayesian finite mixture model: We have K components $1, \dots, K$, distributed as in Subsection 2.3 according to a distribution β, with β itself drawn from a $\mathrm{Dirichlet}(\alpha_1, \dots, \alpha_K)$ distribution.

Each mixture component's parameters are drawn from some prior distribution G_0. Based on the parameter vector ϕ_θ of component $\theta \in \{1, \dots, K\}$, a data point x is generated according to a data model $F(x; \phi_\theta)$. Given the model parameters $\beta, \phi_1, \dots, \phi_K$, the sample $(x_1, \theta_1), \dots, (x_n, \theta_n)$ is generated by choosing for each $i = 1, \dots, n$ a component θ_i and then generating x_i from the component's data model:

1. Choose i's component $\theta_i \sim \beta$
2. Generate $x_i \sim F(\cdot; \phi_{\theta_i})$ where the ϕ_θ are the component parameters, which are themselves distributed according to some prior G_0.

Typically, the Dirichlet hyper-parameters are chosen as a single $\alpha_1 = \dots = \alpha_K = \alpha$, yielding a mean of $1/K$ for all $\beta_1, \dots, \beta_K$. The higher the value of α is chosen, the more uniform the sampled distribution β of the components tends to become, and the more uniform $\beta | \theta_1, \dots, \theta_n$ becomes, analogously to how the posterior distribution of θ given D gets close to 0.5 in our coin flip example if we choose $\alpha_H = \alpha_T \gg |D|$. Picking $\alpha < 1$, on the other hand, leads to sparsity: In contrast to $\alpha > 1$, where the distribution of β has a single mode at its mean $\beta = (1/K, \dots, 1/K)$, it now has K modes at $\beta = (1, 0, \dots, 0), (0, 1, 0, \dots, 0), \dots, (0, \dots, 0, 1)$, favoring component distributions that assign most of the probability mass to a single component.

An application of the mixture model is document clustering based on Latent Dirichlet allocation (LDA) (Blei et al., 2003): Each x is a document represented by its term-frequency vector, a vector whose components are indexed by the words of the vocabulary, and have as value the frequency of the respective word in the document. The data model $F(\cdot; \phi_z)$ of the multinomially chosen components (clusters) z, specifying a distribution over bags of words, i.e., possible documents, is itself multinomial, and therefore it is customary to use the conjugate $\mathrm{Dirichlet}(\alpha', \dots, \alpha')$ prior G_0 over ϕ_z, with the maximum-likelihood interpretation of adding $\alpha' - 1$ pseudocounts for each vocabulary word.

2.5 Dirichlet Processes

One limitation of using the Dirichlet distribution for mixture models is the need to specify the number K of mixture components a-priori. If the true number of components is less than K, the Dirichlet model will learn this and infer the remaining component's probabilities β_i to be close to zero. Consider, however, the case where the number of components is inherently infinite. Fixing the number of mixture components to a specific number K and modeling them with Dirichlet priors would probably not be very elegant, and could still result in under-fitting of the data. Unfortunately, this situation is very common in natural language processing tasks like word segmentation, language modeling, grammar induction, etc. In these situations, a solution can be the use of nonparametric priors.

Dirichlet Processes (Ferguson, 1973) are currently among the most common nonparametric priors. A Dirichlet Process(DP) is a distribution over distributions, with special properties which will be explained below. An example application for the nonparametric model is the task of unsupervised word segmentation (Goldwater et al., 2008), where the goal is to decide about a potential word boundary, and the mixture components are the word types. The DP provides an alternative prior model over the mixture components, with an unbounded complexity. This model allows for introduction of new mixture components as more data is observed. In the word segmentation example, this means that the number of word types increases as more data is seen, and there is always a non-zero probability of adding a new word type.

2.5.1 Definition

Let H be a distribution over event space Θ, and let α be a positive real number. A distribution G over Θ is said to be a draw from a *Dirichlet Process* with base distribution H and concentration parameter α,

$$G \sim DP(\alpha, H) ,$$

if for every measurable partition $A_1, \dots, A_r$ of Θ:

$$(G(A_1), \dots, G(A_r)) \sim \text{Dirichlet}(\alpha H(A_1), \dots, \alpha H(A_r)) \quad (2)$$

2.5.2 The Polya urn construction

In order to understand this definition, consider first the following intuitive urn metaphor for the Dirichlet distribution (Blackwell and MacQueen, 1973). Imagine an urn containing balls of r different colors. Initially the urn contains a possibly fractional count of α_1 balls of color c_1, α_2 balls of color c_2, and so on. We perform N draws from the urn, where after each draw, the ball is placed back into the urn, along with another ball of the same color. In the limit, as the number of draws approaches infinity, the proportions of different colors in the urn will be distributed according to the Dirichlet distribution $\text{Dirichlet}(\alpha_1, \dots, \alpha_r)$.

This metaphor can be extended to explain the DP. We can say that draw $G \sim DP(\alpha, H)$ defines a distribution over partitions of the event space Θ (where Θ is the set of all possible unique colors) as follows: For each measurable partition $A_1, \dots, A_r$ of Θ, the distribution defined by G is the limit of the color proportions of the urn, when the initial count of each color k has been $\alpha H(A_k)$, and, as above, at each step a ball is drawn and placed back into the urn along with a copy of the same color.

In practice, we are usually not interested in G itself, and this distribution is marginalized out. We can also use the urn metaphor without referring to G directly, and model only successive independent draws $\theta_1, \theta_2, \dots$ from G: Imagine that there is an urn that is empty in the beginning. First, color θ_1 is drawn from H, a ball is colored with it, and the ball is added to the urn. In each subsequent step $n+1$, either color θ_{n+1} is drawn from H with probability $\frac{\alpha}{\alpha+n}$ and a ball is colored with it and is added to the urn, or, with probability $\frac{n}{\alpha+n}$, a ball is drawn from the urn, its color is used to color a new ball and both balls will be added to the urn.

In the segmentation example, colors are word types and the balls in the urn are the set of all previous word tokens. For chosing the next word token, with probability $\frac{\alpha}{\alpha+n}$ a new word type is generated. And with probability $\frac{n}{\alpha+n}$, the next word token will take value drawn from the distribution of the previously seen word types.

2.5.3 Properties

Dirichlet Processes have special properties. The base distribution H is the mean of the Dirichlet Process, and for any measurable $A \subseteq \Theta$, we have $E[G(A)] = H(A)$. Also, the concentration parameter, α, can be understood as an inverse variance:

$$V[G(A)] = \frac{H(A)(1 - H(A))}{\alpha + 1}$$

The larger α becomes, the more the Dirichlet Process concentrates its mass around the mean. As α goes to infinity, we will have $G(A) \rightarrow H(A)$ for any measurable set A.

In the word segmentation example, Θ is the set of all possible words in the language. H is a base distribution over Θ, and $DP(\alpha, H)$ is a distribution over possible distributions over Θ. A draw from the DP, $G \sim DP(\alpha, H)$, is one distribution over Θ, with the α parameter deciding how close G is to H for each word type. As we will see in the next sub-section, a smaller value of α means a lower tendency to add new word types.

Given a sequence of independent samples, $\theta_1, \cdots, \theta_n$ from G, the posterior distribution of G given $\theta_1, \cdots, \theta_n$ can be calculated as a direct result of the conjugacy between Dirichlet and multinomial distributions:

$$G \mid \theta_1, \cdots, \theta_n \sim$$

$$DP(\alpha + n, \frac{\alpha}{\alpha + n} H + \frac{n}{\alpha + n} \frac{\sum_{i=1}^{n} \delta_{\theta_i}}{n})$$

where δ_{θ_i} is a distribution concentrated at θ_i, i.e., $\delta_{\theta_i}(x)$ is the identity function $I(x = \theta_i)$, and $\frac{\sum_{i=1}^{n} \delta_{\theta_i}}{n}$ is the empirical distribution of draws from event space Θ.

Using this formula we can compute the predictive probability of θ_{n+1}, conditioned on the previous draws $\theta_1, \dots, \theta_n$. For each measurable set $A \in \Theta$

$$\begin{aligned} P(\theta_{n+1} \in A \mid \theta_1, \cdots, \theta_n) \\ = E[G(A) \mid \theta_1, \cdots, \theta_n] \\ = \frac{1}{\alpha + n} (\alpha H(A) + \sum_{i=1}^{n} \delta_{\theta_i}(A)) \quad (3) \end{aligned}$$

where $\delta_{\theta_i}(A)$ is one if $\theta_i \in A$ and zero otherwise.

The $\text{Dirichlet}(\alpha_1, \cdots, \alpha_K)$ distribution is a special case of $DP(\alpha, H)$ with $\Theta = \{1, \cdots, K\}$, $\alpha = \sum_{k=1}^{K} \alpha_k$ and $H(\{k\}) = \alpha_k/\alpha$ for all $k = 1, \ldots, K$. Note how Equation 3 for $A = \{k\}$ then coincides with Equation 1 from Subsection 2.3, since $H(A) = \alpha_k/\alpha$ and $\sum_{i=1}^{n} \delta_{\theta_i}(A)$ is the total number of samples with outcome k, denoted n_k in Equation 1.

2.5.4 Chinese Restaurant Process Representation

An alternative metaphor for the urn scheme construction of the posterior distribution of the DP (cf. Subsection 2.5.2) is the Chinese Restaurant Process (CRP) representation, which will also turn out useful for understanding the generalizations of the DP later on. Consider a Chinese restaurant with an unbounded number of tables and unbounded seating capacity for each table. Customers i enter the restaurant one by one, choose to sit at a table z_i based on the seating arrangement of the previous customers, and order a dish $\theta_i \in \Theta$. In the CRP, the probability of customer $i + 1$ choosing table $z_{i+1} = k$ is

$$P(z_{i+1} = k \mid z_1, \ldots, z_i) = \begin{cases} \frac{\sum_{j=1}^{i} I(z_j = k)}{i+\alpha} & 1 \leq k \leq K(z_1, \ldots, z_i) \\ \frac{\alpha}{i+\alpha} & k = K(z_1, \ldots, z_i) + 1 \end{cases} \tag{4}$$

where $K(z_1, \ldots, z_i)$ is the total number of tables occupied by the i customers so far.

If customer $i + 1$ chooses to sit at an existing table k ($1 \leq k \leq K(z_1, \ldots, z_i)$), he will order the same dish θ_{i+1} as others on that table. If he chooses to sit at a new table $k = K(z_1, \ldots, z_i) + 1$, he will order a dish from base probability distribution H over all possible dishes. The so constructed $\theta_1, \ldots, \theta_n$ can now be interpreted as draws from a hidden distribution G, which itself was drawn from a $DP(\alpha, H)$.

Let $\phi_1, \ldots, \phi_K \in \Theta$ denote the dishes being served at tables $1, \ldots, K$ (note that identical dishes can be served at different tables). In the word segmentation example, the set Θ of all possible dishes is the set of word types in the language. Each table is a mixture component. Each customer i corresponds to a word position in the text, and his dish θ_i to the word at position i. Based on Equation (4), the next word $i + 1$ takes the form ϕ_k with probability proportional to the number of previous word tokens having already taken the form ϕ_k. And with probability $\frac{\alpha}{i+\alpha}$, it is associated with a new mixture component, and the mixture label ϕ_{K+1} is drawn from a base distribution over word types.

2.5.5 Stick-breaking Construction

Yet another way to characterize the Dirichlet Process is by the *stick-breaking construction.* As shown by Sethuraman (1994), a distribution $G \sim DP(\alpha, H)$ can be constructed as follows:

$$\begin{aligned} \beta_k &\sim \text{Beta}(1, \alpha) \\ \pi_k &= \beta_k \pi_{l=1}^{k-1} (1 - \beta_l) \\ \theta_k^* &\sim H \\ G &= \sum_{k=1}^{\infty} \pi_k \delta_{\theta_k^*} \end{aligned}$$

The distribution over $\pi = (\pi_1, \pi_2, \ldots)$ is commonly referred to as the *stick-breaking distribution* and denoted:

$$\pi \sim \text{GEM}(\alpha)$$

We can imagine its construction as breaking a length-one stick at ratio β_1, assigning π_1 the length of the broken-off part, and recursively breaking the remaining stick at ratio β_k to obtain π_k.

2.5.6 Dirichlet Process Mixture Models

The most common application of DPs is the use in mixture modeling (Antoniak, 1974). Unlike other Bayesian models, nonparametric models allow for an unbounded number of mixture components. For these problems, input is usually a sequence of observations $x_i, \ldots, x_n$, generated from latent variables $\theta_1, \ldots, \theta_n$. Each θ_i is an independent draw from G, and each x_i is drawn from the distribution $F(\theta_i)$.

$$G \mid \alpha, H \sim DP(\alpha, H)$$

$$\theta_i \mid G \sim G$$

$$x_i \mid \theta_i \sim F(\theta_i)$$

The word segmentation task is an example of this application of the DP, where the word tokens are hidden variables, θ_i, drawn from the infinite mixture components, and observations are character sequence of x_is. Given the observations, the posterior inference on the hidden variables can be performed by Markov Chain Monte Carlo (MCMC) sampling procedures such as Gibbs sampling. For some applications, the sampler can be designed to be tractable (DeNero et al., 2008).

2.6 Hierarchical Dirichlet Processes

Dirichlet Processes provide a solution to the component number selection in mixture models. However, in many settings, especially in the area of natural language processing, it is often the case that the mixture components have latent dependencies between themselves. For instance, in n-gram language models, the distribution of the next word depends on the previous n-1 word choices. As another example, in inferring the derivation tree from a word sequence under a context free grammar model, the choice of each nonterminal affects the choices for the re-write selections. These inter-component dependencies are captured by *Hierarchical Dirichlet Process (HDP)* models (Teh et al., 2006).

Depending on the application, HDP models can introduce different dependency networks (Teh et al., 2006). A simple way to add hierarchy of mixture models is to assume one underlying $DP(\alpha_0, G_0(\tau))$, and consider each mixture model G_j to be a conditionally independent draw from the base DP.

$$G_j \mid \alpha_0, G_0(\tau) \sim DP(\alpha_0, G_0(\tau))$$

Although this model allows clusters to arise within each mixture component, in the case where $G_0(\tau)$ is a continuous distribution, the G_js do not necessarily have any atoms in common, and thus, different mixture components can not share clusters between themselves. However, if G_0 itself is a draw from another DP then we have a discrete base distribution, and in that case the mixture components can share atoms.

There are other notions of the HDP. For instance, Muller et al. (2004) propose that each group is an interpolation of draws from two DPs: $G_j = \varepsilon F_0 + (1 - \varepsilon)F_j$. This formulation imposes a common distribution on clusters within all group. However, in NLP applications we need to have a $partial$ sharing of clusters between groups so that the power-law property can be imposed on the shared clusters as well. Thus, in this review we will only focus on the model proposed by Teh et al. (2006).

2.6.1 Definition

A hierarchical Dirichlet process is a distribution over a set of random probability measures over (Θ, β). The process defines a set of random probability measures G_j, one for each group, and a global random probability measure G_0. The global measure G_0 is distributed as a DP with concentration parameter γ and base probability measure H:

$$G_0 \mid \gamma, H \sim DP(\gamma, H)$$

The random measures G_j are conditionally independent given G_0, with distributions given by a Dirichlet process with base probability measure G_0:

$$G_j \mid \alpha_j, G_0 \sim DP(\alpha_j, G_0)$$

The hyper-parameters of the hierarchical Dirichlet process consist of the baseline probability measure H, and the concentration parameters γ and α_j. This model can be extended to more than two levels, and the formulas can be easily derived for additional levels.

2.6.2 Chinese Restaurant Franchise

The CRP metaphor used for the Dirichlet process can be extended to explain the HDP as well. The extension introduces the concept of a Chinese restaurant franchise, in which multiple restaurants share the same menu.

Each restaurant corresponds to one group, (i.e. left-hand side nonterminal symbol in the CFG derivation, or the previous word in the bigram language model). Each table in the restaurant corresponds to clusters within group (i.e. values that rule type can take, or values that the second word can take in bigram language model), and customer corresponds to factors (i. e. next rule type choice, or second word choice in the bigram language model). Based on this definition, probability of i^{th} customer sitting at table which has dish ψ_{jt} in restaurant j is calculated by first integrating out G_j:

$$\theta_{ji} \mid \theta_{j,i-1}, \ldots, \theta_{j,1} \sim$$
$$\sum_{t=1}^{m_{j.}} \frac{n_{jt.}}{i-1+\alpha_j} \delta_{\psi_{jt}(\theta_{ij})} + \frac{\alpha_j}{i-1+\alpha_j} G_0(\theta_{ij})$$

where $m_{j.}$ means total number of tables occupied in restaurant j, and $n_{jt.}$ represents count of customers seated at restaurant j and table t. Like before, δ is the identity function. This formula means that each customer i in restaurant j either joins a table that serves ψ_{jt} with probability proportional to the fraction of customers seated at tables with dish ψ_{jt}, or sits at a new table and orders a dish from G_0 with probability $\frac{\alpha_j}{i+1+\alpha_j}$.

At the next level,G_0 can also be integrated out, leading to

$$\psi_{jt} \mid \psi_{1,1}, \ldots, \psi_{2,1}, \ldots, \psi_{j,t-1}, \gamma, H \sim$$
$$\sum_{k=1}^{K} \frac{m_{.k}}{m_{..} + \gamma} + \frac{\gamma}{m_{..} + \gamma} H(\psi_{jt})$$

where $m_{..}$ is the total count of tables occupied in the franchise.

To obtain samples of θ_{ji} we first sample them according to $P(\theta_{ji} \mid \theta_{j,i-1}, \ldots, \theta_{j,1})$, and if a new sample from G_0 is needed, we use the above equation to obtain a new sample ψ_{jt} and set $\theta_{ji} = \psi_{jt}$.

2.6.3 Problem Setting and Inference

HDP nonparametric models have been extensively used in several Natural Language Processing tasks, because on one hand, they provide capability to model shared properties among unbounded mixture components. On the other hand, this added power does not cognitively complicate the model's inference procedures. The MCMC samplers usually become computationally more expensive though. HDP models have been used for PCFG induction, synchronous grammar induction, alignment and phrase extraction, and context based segmentation. We will see some examples of HDP models more closely in Section 3.

2.7 Pitman Yor

The Pitman Yor Process (Pitman and Yor, 1997) is another extension of the Dirichlet Process, which adds a discount parameter d to $DP(\alpha, H)$ to allow the nonparametric model to have more control over the increase rate in the number of mixture components. A draw G from a Pitman-Yor is denoted as:

$$G \sim \mathrm{PY}(d, \alpha, H)$$

Following the urn metaphor, in the DP model, as the number n of observed data points increases, the probability of a value coming from H decreases at the fixed rate of $\frac{\alpha_0}{n+\alpha_0}$. Pitman Yor processes change this probability to $\frac{\alpha_0+d\times t}{n+\alpha_0}$,with t being the total number of draws from H so far and $0 <= d < 1$ being the discount parameter of the model. Compared to the DP, the Pitman Yor process can have higher component growth rate, and in fact, the number of unique values (e.g. words in a language model application) generated by this model scales as $O(\alpha_0 n^d)$ when $d \neq 0$. When $d = 0$, Pitman Yor reduces to the Dirichlet Process, with a new value generation rate of $O(\alpha_0 \log n)$.

Pitman-Yor processes can thus produce power-law distributions, and therefore have been argued to be more suitable to applications in natural language processing Goldwater et al. (2006).

3 Applications

In this section we will examine applications of each of the models discussed in Section 2. We will first discuss the task of transcribed speech segmentation using DP and HDP models. The next application is bilingual phrase extraction and alignment, in which the authors propose two models based on DP and HDP. We will then review the task of context-free parsing based on the HDP model, and finally we will give an overview of the Hierarchical Pitman-Yor Process model used for language modeling.

3.1 Word Segmentation with DP and HDP

Goldwater et al. (2008) apply the DP and HDP models to the task of transcribed speech segmentation. They define two models. The first model is a Unigram model based on a DP prior. The advantage of using DP is that the number of word types is not assumed to be finite, and at each point, the probability of viewing a new word is always non zero.

3.1.1 Unigram Generative Model

The unigram model consists of generating sentences independently, and in each sentence, generating words independently with probability $(1 - p\$)P(w)$, and generating the end of sentence mark with probability of $p\$$. Each word is drawn from a Dirichlet Process prior $DP(\alpha_0, P_0)$, which basically means that for each word, it is first decided

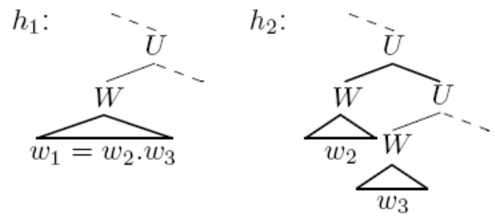

Figure 1: Hypotheses h_1 and h_2 to sample from by Gibbs Sampler.

if it is a novel word, and if yes, the word is from the base distribution P_0. And if it is not novel, it is chosen from the current lexicon:

$$P(w_i = l \mid w_{i-1}, \dots, w_1) =$$

$$\frac{n_l}{i-1+\alpha_0} + \frac{\alpha_0 P_0(w_l = l)}{i-1+\alpha_0}$$

Where n_l is the number of w_js that have taken the value l before.

The authors define the base distribution as a unigram model over phonemes:

$$P_0(w_l = x_1 \cdots x_M \mid w_i \; is \; novel) =$$

$$p_{stop}(1 - p_{stop})^{M-1} \prod_{j=1}^{M} P(x_j)$$

Where p_{stop} is the probability of stop, and $P(x_j)$ is a uniform distribution over phonemes.

3.1.2 Inference for Unigram Model

The generative model allows to compute the probability of any segmentation of a given input, and in order to infer the best segmentation, a Gibbs sampler is defined. The Gibbs sampler considers each potential segmentation boundary in text, (i.e. sampler's variables are boundary positions). At each potential boundary, the sampler considers two hypotheses h1 in which the segmentation boundary is set to true, and h2 in which the segmentation boundary is set to false. Figure 1 shows h1 and h2.

The Gibbs sampler's transition probabilities can be calculated from the generative model's DP and Base model formulas. The sampler is run for 20,000 iterations, and outputs the sampler's state at iteration 20,000 as the final segmentation.

3.1.3 Bigram Generative Model

The authors then define a bigram model to incorporate contextual dependencies in their segmentation task. As mentioned in section 2.3, hierarchical Dirichlet processes are one extension of DPs which allow shared factors among mixture components. The bigram model defined in this generative model contains HDP prior which defines one DP per word.

The model proceeds by choosing each word, conditioned on the previous word, until the sentence end mark is generated. Going back to the Chinese Restaurant Franchise metaphor, each customer (i.e. a word) first decides whether to go to a restaurant with customers (i.e. be second part of part an existing bigram), or choose an empty restaurant (i.e. be part of a new bigram). If it enters a restaurant with customers, the customer then decides which table to sit based on the Dirichlet process of that restaurant. If it goes to a new restaurant, it then proceeds to sit on an empty table and orders dish d (i.e. be of a word type d) with the base probability $P_0(d)$. After integrating out the HDP prior, the predictive probability of the word w_i given the previous words $w_{i-1}, \dots, w_i$ is:

$$P(w_i \mid w_{i-1}, \dots, w_1) =$$

$$\frac{n_{\langle w_{i-1}, w_i \rangle} + \alpha_1 \frac{t_{w_i} + \alpha_0 P_0'(w_i)}{t + \alpha_0}}{n_{w_{i-1}} + \alpha_1}$$

Where $n_{\langle w_{i-1}, w_i \rangle}$ is the number of bigrams $\langle w_{i-1}, w_i \rangle$ previously seen, and t_{w_i} is total number of bigrams starting with w_i, and t is the total number or bigrams. And since stop mark is also generated depending on the previous word, the model defines $P_0'(w_i)$ which is $p\$$, if $w_i = \$$, and is $(1 - p\$)P_0(w_i)$ for all other w_is. P_0 is the same as in unigram model.

3.1.4 Inference for the Bigram Model

The inference for bigram model is performed similarly, by a Gibbs sampler which moves over all segmentation positions and selects the segmentation variable (true/false) at each location, and after enough iterations, it converges to the correct segmentation. The only note about the sampler for bigram model is that in deciding between $s1 = \beta w_1 \gamma$ and $s2 = \beta w_2 w_3 \gamma$ one word to the left and one word to the right of the w_1 and $w_2 w_3$ should also be considered in the formulations.

Model	F_0 (tokens)	F_0 (types)
Unigram-based		
NGS-u	68.9	52.0
MBDP-1	68.2	52.4
DP	53.8	57.2
Bigram-based		
NGS-b	68.3	55.7
HDP	72.3	59.1

Table 1: Word segmentation accuracy of unigram and bigram systems.

3.1.5 Results and Discussions

The two models are tested on the CHILDES corpus, which contains 9790 sentence, 33399 word tokens, and 1321 word types. The average number of words per sentence is 3.4, and average word length (in Phonemes) is 2.48. Evaluation metrics are Precision, Recall and F measure on word tokens and on word types, and also on potentially ambiguous boundaries.

Unigram model has two hyper-parameters, p_{stop} and α_0. Lower values for p_{stop} results in under-segmentation, in which longer words are preferred. Higher α_0 results in a higher novel word type generation rate, which leads to segmentation of long chunks into shorter words. Given this trade-off where high novel word rate improves Recall on word types and decrease the precision on word Tokens, there is no single best value for p_{stop} and α_0. Thus, the authors have simply fixed these two parameters to $p_{stop} = 0.5$ and $\alpha_0 = 20$.

They compare their result with two other segmentation models: N-Gram Segmentation method(NGS) (Venkataraman, 2001), and Model Based Dynamic Programming(MBDP)(Brent, 1999). NGS finds the maximum likelihood solution, when in fact, optimal solution is the unsegmented text in theory and its result are based on the search procedure rather than correct model. MBDP method tries to remove this problem by assuming a prior over segmentation hypothesis, but is not capable of handling n-gram dependencies. Some of the result of the methods are shown in Table 1.

The results confirm that the performance of the maximum likelihood approach comes from its approximate search strategy rather than the likelihood function definition. As bigram dependencies are introduced to the models, it can be seen that improvement of HDP over the DP is more significant than the improvement of NGS-bigram over NGS-unigram. Also, based on the results, type accuracy in DP model is better than token accuracy, which indicates many errors on the frequent words. But in the HDP model, the opposite behavior is observed, showing that frequent words are more likely to be segmented correctly by the HDP model.

3.2 Alignment and Phrase extraction with DP and HDP

In phrase based translation models, phrase translation information is usually achieved by inferring the alignment, and collecting phrase translation pairs by a set of heuristics. The computation of alignment expectations under general phrase models is a P-hard problem, and this has prevented the extensive use of probabilistic phrase alignment models. Another problem in learning phrase alignment is potential degeneracy, which is a common theme in any maximum likelihood model. At a position to choose between large and small structures, the maximum likelihood models prefer the large structures. DeNero et al. (2008) propose two new models based on DPs and HDPs to solve the problem of degeneracy, and also design a tractable sampling algorithm to compute the phrase count expectations for phrase pairs.

3.2.1 Generative Model

Each sentence pair in the corpus is generated independently. To generate one sentence pair, a number of (l) independent phrase pairs are generated, and these phrases are then reordered on the foreign side. The ordering of phrases on the English side is assumed to be fixed by generation order.

In order to model cases where phrase pairs contain non-equal information, each phrase pair can be null aligned with probability P_ϕ. And conditioned on whether it is null aligned, or not, the phrase pair is then drawn from θ_N and θ_J accordingly. θ_N is a multinomial distribution that generates the null aligned phrases on both source and target side. θ_J is a DP that generates pairs of non-null phrases.

Based on this model, the joint probability of phrase segmentation and alignment is calculated:

$$P(\{\langle e,f\rangle\},a) =$$

$$p_{\$}(1-p_{\$})^{l-1}P(a \mid \{\langle e,f\rangle\}) \prod_{\langle e,f\rangle} P_M(\langle e,f\rangle)$$

Where

$$P_M(\langle e,f\rangle) =$$

$$P_\phi \theta_N(\langle e,f\rangle) + (1-P_\phi)\theta_J(\langle e,f\rangle)$$

Probability of alignment given phrase pairs is simply defined as a product of position based distortion penalties for each phrase pair.

Because of the unbounded space of phrase pairs, nonparametric modeling is used to model θ_J. The authors define a DP prior for the non-null phrase pairs, and try to reduce the degeneracy problem by making the base distribution of this Dirichlet prior to give higher probabilities to shorter phrases. They use IBM Model 1 probabilities to incorporate the prior knowledge into the model:

$$\theta_J \sim DP(\alpha, M_0)$$

$$M(\langle e,f\rangle) =$$

$$[P_f(f)P_e(e)P_WA(f \mid e)P_WA(e \mid f)]^{\frac{1}{2}}$$

Also, $P_e(e)$ and $P_f(f)$ are defined in a way to encourage shorter phrases:

$$P_e(e) = P_G(|e|; P_s).(\frac{1}{n_e})^{|e|}$$

$$P_f(f) = P_G(|f|; P_s).(\frac{1}{n_f})^{|f|}$$

While DP is a good model for non-null phrases, it is not a good choice for null-aligned phrase pairs. The reason is that the DP imposes a rich-get-richer property over the phrase pair distributions, and common words that has had null alignments at first tend to align with null more and more. For this reason, the null aligned distribution was defined as a unigram model, uniform over word types.

$$\theta_N(\langle e,f\rangle) = \begin{cases} \frac{1}{2}.P_e(e) & \text{if e=null} \\ \frac{1}{2}.P_f(f) & \text{if f=null} \end{cases}$$

3.2.2 Collapsed Gibbs Sampling

After the definition of the model, the paper introduces a novel Gibbs sampler. The sampler will sample over all possible phrase segmentations at source and target side, as well as the alignment links between the source and target phrases. To do exact inference, computing the alignment expectations under a general phrase model is a #P-hard problem(DeNero and Klein, 2008). But the sampler proposed in this paper uses polynomial-time computable operators to calculate the approximation of phrase alignment expectations.

The authors define two sampler operators SWAP and FLIP, that together form a complete sampler. SWAP operator creates new states by swapping alignment links between two phrase pairs. SWAP operator can arbitrarily shuffle the phrase alignments but another operator is needed to actually change the phrase boundaries. FLIP operator is responsible for changing the status of one single segmentation position at each step. Transition probability for each of these operators can be calculated based on the DP prior for θ_J. The collapsed sampler integrates out this prior by replacing it with the Chinese Restaurant Process representation of the DP:

$$P(\langle e,f\rangle \mid z_m) =$$

$$\frac{count_{\langle e,f\rangle}(z_m) + \alpha.M_0(\langle e,f\rangle)}{|z_m| + \alpha}$$

Where z_m is the set of all non-null phrases observed in the fixed part of the hypothesis. Probabilities of each operator's two possible outcomes are then calculated accordingly. A degeneracy analysis of the size of the sampled phrases confirms that this model yields a non-degenerate distribution over phrase lengths.

To estimate the expected phrase counts, after each iterations(after each operator has been applied to every position at each sentence), the counts are accumulated.

$$\frac{1}{N}\sum_{i=1}^{N} count_{\langle e,f\rangle}(x, z_i) \longrightarrow E[count_{\langle e,f\rangle}(x,.)]$$

3.2.3 HDP Model

As a generalization of the DP model, this paper also defines a hierarchical Dirichlet process prior for θ_J. Instead of defining one DP from which

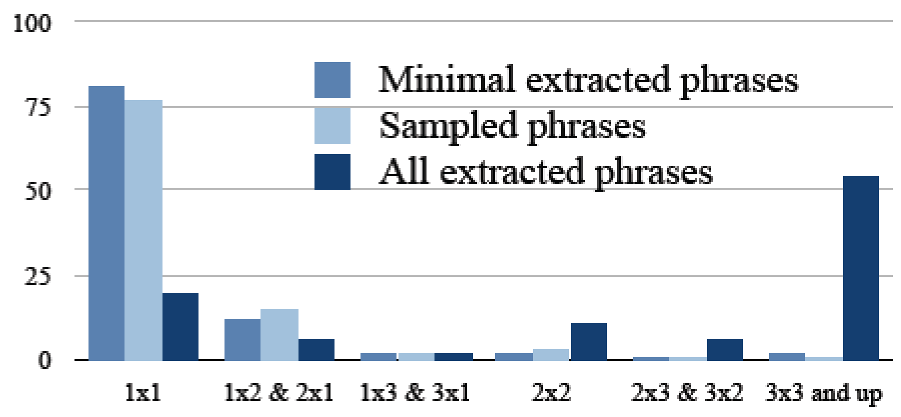

Figure 2: From DeNero et al. (2008): Distribution of phrase pair sizes, denoted as English phrase size $\times$ Foreign phrase size.

Phrase extr. model	NIST-BLEU
Baseline (‘grow-diag-final-and’)	29.8
DP	30.1
HDP	30.1

Table 2: Translation accuracy (NIST-BLEU score) of standard-SMT phrase extraction based on the ‘grow-diag-final-and’ method vs. Dirichlet-Process and Hierarchical-Dirichlet-Process based phrase extraction models.

phrase pairs are drawn, in the HDP model, two languages E, and F each have an independent DP prior over the monolingual phrases. The pairs are then derived from these phrases based on an HDP process.

$$\theta_j \sim DP(\alpha, M'_0)$$

$$M'_0(< e, f >) =$$

$$\theta_F(f)\theta_E(e)[P_W A(f \mid e) P_W A(e \mid f)]^{\frac{1}{2}}$$

3.2.4 Results and Discussions

The experiments have been done for Spanish to English translation task, with the baseline system trained on Europarl sentences of length up to 20. The sampling was done for 100 iterations, and the initialization of the sampler is based on the word alignments generated by the baseline. For the DP model, the result has been independent of α parameter, and it has been fixed to 100.

Table 2 shows the result of the two nonparametric models compared to the baseline. Both DP and HDP based models achieve a modest improvement over the heuristic-phrase-extraction baseline.

One possible problem could be that the sampler had been run for 100 iterations only, and the paper does not provide any information about the convergence of the sampler at this number of iterations. Compared to the segmentation task whose sampler has taken 20,000 iterations, these experiments seem unconvincing on the convergence side. Note, however, that the number of training instances is several orders of magnitudes larger compared to the segmentation application, so one would expect this model to converge earlier in terms of number of iterations than the segmentation model.

3.3 Context-free Parsing with HDP Priors

Most state-of-the-art parsers, as e.g. the one by Charniak and Johnson (2005) for the task of treebank parsing, are based on Probabilistic Context-Free Grammars (PCFGs). Although the final task in treebank parsing is producing the correctly-labeled phrase-structure tree for a given sentence, the nonterminals of the PCFG grammar learned by state-of-the-art parsing models tend to be more complicated than the treebank labels (e.g., parent-annotated), giving rise to PCFG trees with labels that are more complicated than the original treebank labels, but from which these labels can easily be recovered. This technique is called *grammar refinement* and has been applied e.g. by Collins (2003) and Klein and Manning (2003). Determining how fine-grained the model should be can be a challenge: The more grammar symbols a treebank label is split into, the more powerful the grammar becomes at capturing linguistic dependencies occuring in the training data, but at the same time it becomes more prone to overfitting.

3.3.1 The infinite PCFG

Liang et al. (2007) address this problem directly by equipping an infinite-nonterminal-set PCFG with a Hierarchical Dirichlet Process prior, which regularizes the number of nonterminal symbols effectively used in the estimated grammar. The model, which they call HDP-PCFG, is based on PCFGs in Chomsky Normal Form, i.e., grammar rules can be of either of the following two forms:

$$A \rightarrow BC$$

$$A \rightarrow a$$

where $A, B, C \in \mathcal{N}$ are nonterminals and $a \in \mathcal{T}$ is a terminal symbol. We can thus think of a PCFG derivation in terms of transitions (expressed by the former type of rule) and emissions (the latter) as for Hidden Markov Models, except that each derivation step makes a choice between either a transition *or* an emission, and that a transition forks off into two next states instead of a single one.

The generative story of the growth of an HDP-PCFG forest now goes as follows: First, a top-level distribution β over the countably infinite set of grammar nonterminals $\mathcal{N} = \{1, 2, \ldots\}$ is drawn according to the stick-breaking DP representation (cf. Subsection 2.5.5):

$$\beta \sim \mathrm{GEM}(\alpha)$$

Now, for each nonterminal symbol $z \in \{1, 2, \ldots\}$, we draw its rule type probabilities $\phi_z^T \in [0, 1]^2$ (emission vs. transition), its emission probabilities $\phi_z^E \in [0, 1]^{\mathcal{T}}$ (one for each terminal symbol), and its binary-production transition probabilities $\phi_z^B \in [0, 1]^{\mathcal{N}^2}$ (one for each possible pair of nonterminals $\langle z_L, z_R \rangle \in \{1, 2, \ldots\}^2$, expressing the probability of z rewriting into z_L, z_R):

$$\phi_z^T \sim \mathrm{Dirichlet}(\alpha^T)$$

$$\phi_z^E \sim \mathrm{Dirichlet}(\alpha^E)$$

$$\phi_z^B \sim DP(\alpha^B, \beta\beta^T)$$

where $\beta\beta^T$ is the outer product, resulting in a doubly-infinite matrix whose elements sum up to one. This matrix, itself based on the outcome of a Dirichlet Process, now serves as the base distribution for the inner Dirichlet Process determining the production probabilities.

K	F_1 (PCFG)	F_1 (smoothed PCFG)	F_1 (HDP-PCFG-GR)
1	60.47	60.36	60.5
2	69.53	69.38	71.08
4	75.98	77.11	77.17
8	74.32	79.26	79.15
12	70.99	78.8	78.94
20	64.44	79.27	77.81

Table 3: Parsing scores of PCFG models vs. HDP-PCFG-GR for the scarce-date scenario. The truncation level K determines the models' maximum number of subsymbols per treebank symbol.

The generative story continues by drawing for each tree to be generated, and for each such tree's node i to be generated, from the prior distributions above as follows ($z_i \in \mathcal{N}$ denotes the label of i, with z_1 being the start symbol of the grammar; $L(i)$ and $R(i)$ denote yet-to-be generated left and right child nodes of i):

```
$t_i \sim \phi^T_{z_i}$
If $t_i =$'EMISSION':
    $x_i \sim \phi^E_{z_i}$
Else:
    Add new nodes $L(i), R(i)$ to tree
    $\langle z_{L(i)}, z_{R(i)} \rangle \sim \phi^B_{z_i}$
```

3.3.2 Tackling Grammar Refinement

In the grammar-refinement scenario, each treebank symbol s is refined into a set $\mathcal{N}_s$of subsymbols, and the resulting nonterminal set of the PCFG is the disjoint union of all subsymbol sets. To remain uncommitted about the number of subsymbols to allocate for a given treebank symbol s, we keep each $\mathcal{N}_s$ countably infinite, and draw its distribution β_s given s from a DP. The individual treebank-symbol specific DP distributions are tied together in the binary production generation step: The base distribution of its DP is now an outer product between distribution vectors $\beta_{s'}$ and $\beta_{s''}$ from generally different symbol-specific distributions. The complete generative model for this

Grammar-Refinement HDP-PCFG variant, dubbed HDP-PCFG-GR is given below:[1]

```
For each treebank symbol $s$:
  $\beta_s \sim \mathrm{GEM}(\alpha)$
  For each subsymbol $z \in \{1, 2, \ldots\}$:
    $\phi^T_{sz} \sim \mathrm{Dirichlet}(\alpha^T)$
    $\phi^E_{sz} \sim \mathrm{Dirichlet}(\alpha^E(s))$
    $\phi^b_{sz} \sim \mathrm{Dirichlet}(\alpha^b)$
  For all treebank symbols $s', s''$:
    $\phi^B_{szs's''} \sim DP(\alpha^B, \beta_{s'}\beta^T_{s''})$
For each parse tree node $i$:
  $t_i \sim \phi^T_{s_i z_i}$
  If $t_i =$'EMISSION':
    $x_i \sim \phi^E_{s_i z_i}$
  Else:
    Add new nodes $L(i), R(i)$ to tree
    $\langle s_{L(i)}, s_{R(i)} \rangle \sim \phi^b_{s_i z_i}$
    $\langle z_{L(i)}, z_{R(i)} \rangle \sim \phi^B_{s_i z_i s_{L(i)} s_{R(i)}}$
```

3.3.3 Experiments and Discussion

Liang et al. (2007) first empirically evaluate their HDP-PCFG-GR model on the task of recovering a simple synthetic grammar from a treebank with hidden dependencies that can be resolved by grammar refinement: Given an original PCFG grammar with one start symbol S and four regular nonterminal symbols $X_1, \ldots, X_4$, 2000 trees are sampled, and then all four regular nonterminals are collapsed into one. Given the resulting treebank, a baseline PCFG model as well as the HDP-PCFG-GR model are trained to recover the refined grammar productions. Whereas the baseline PCFG model, allowed to use 20 subsymbols per tree symbol, spreads the probability mass roughly equally over all subsymbols, thereby creating a huge number of nonzero-probability grammar rules, the HDP-PCFG-GR model recovers the original grammar nearly perfectly.

While this is certainly a nice result demonstrating the merits of the HDP-PCFG-GR model, it could be argued that it is not the 'job' of the PCFG model to find the simplest explanation for the training trees it is presented, but simply to find a model maximizing their joint probability and thus to be able to distinguish them from trees that could not have come from the original grammar. It therefore would have been interesting to see if the learned PCFG grammar generates the same treebank (i.e., nonterminal-collapsed) trees as the original grammar, with the same probabilities. Consider for example a potentially learned model that has five copies of the non-startsymbol rules of the original grammar, which differ only in the nonterminal labels $X_1, \ldots, X_4$, which are mapped to $X_{1\times j}, \ldots, X_{4\times j}$ in the j-th copy. By further splitting the four start symbol rules of their synthetic grammar into 4×5 rules, one for each nonterminal copy, and with $1/5$ of the original weight, we arrive at a larger grammar using all 20 X subsymbols, but nevertheless producing the same treebank with the same frequencies as the original grammar.

More interesting is the second empirical evaluation, in which the HDP-PCFG-GR model is applied to parsing the Peen Treebank. To demonstrate how Bayesian modeling can be an effective guard against overfitting, an ordinary grammar-refinement PCFG estimated with maximum likelihood Matsuzaki et al. (2005) is compared to HDP-PCFG-GR for a scarce-resource task simulated by training only on Section 2 of the Penn treebank instead of the normally used combined Sections 2-21. As additional baseline model, a variant of the grammar-refinement PCFG smoothed with a non-refined PCFG model is provided. Table 3 shows the resulting parsing accuracy for different levels K of subsymbols per treebank symbol. Whereas the PCFG clearly overfits for values of $K > 4$, HDP-PCFG-GR increases its performance with increasing K up to $K = 8$. For values higher than $K = 8$, it starts to slightly deteriorate. Unfortunately (or fortunately for the pragmatist favoring the simple and fast), the smoothed PCFG is able to gain just the same parsing accuracy when increasing K up to 8, and shows even less deterioration than HDP-PCFG-GR when K is increased further. In a full-scale experiment with models trained on all 20 treebank sections, the smoothed PCFG model obtains an F_1 score of 88.36, clearly outperforming HDP-PCFG-GR, which obtained a score of 87.08. On a positive (or negative) note, the number of rules of the trained HDP-PCFG-GR grammar tends to be around 30-50 percent lower compared to the smoothed PCFG.

[1]Liang et al. (2007) also allow the case of unary rules, which we have omitted here for the sake of simplicity.

In conclusion, nonparametric PCFG models can be applied to real-world parsing tasks, and overcome the overfitting problem that PCFGs trained by maximum-likelihood have. While estimating the choice of the right number of parameters based on the available training data in a principled way, they have not yet been shown to actually outperform simple hacks tackling the same problem.

3.4 Language Modeling based on Hierarchical Pitman-Yor Processes

The task of language modeling is to assign a probability distribution to the set of all possible utterances in a certain language and domain. It has a wide range of applications, for instance in speech recognition, machine translation, and spelling correction, to mention but a few. Most state-of-the-art models fall into the class of n-gram models, modeling the probability of an utterance of words $w_1, \ldots, w_M$ (where w_1 is usually a special start symbol and w_M a special stop symbol) as:

$$\begin{aligned} P(w_1^M) &= \prod_{i=1}^{M} P(w_i | w_1^{i-1}) \\ &\approx \prod_{i=1}^{M} P(w_i | w_{i-n+1}^{i-1}) \end{aligned}$$

Maximum-likelihood estimation for these models essentially amounts to counting chunks of n words (*n-grams*) in a training corpus of utterances. For any training corpus size, the choice of n is a dilemma: The higher the value of n, the more accurate the approximation above becomes, and the more powerful our model gets at capturing longer-distance dependencies between words. At the same time, the higher the value of n, the more prone the model becomes to overfitting the training data, and the higher the expected number of unknown events in testing utterances becomes, as more and more sequences of n-grams simply will not have occurred during training. This problem is amplified in the case of human language, whose words are distributed roughly according to a power law, with a lot of word types occurring only very rarely. The most common solution to this problem is smoothing, where multiple k-gram models for $k = 1, \ldots, n$ are combined together by either back-off or interpolation.

Teh (2006) proposes a nonparametric Bayesian approach to this problem, with the aim of a principally justified framework, and of understanding more about why and how certain methods work better than others. The model, based on the hierarchical Pitman-Yor Process, is a generalization of an HDP language model previously proposed by Mackay and Petoy (1995).

3.4.1 Model definition

Remember that an n-gram language model assigns a probability to a word given a context $u = u_1 \cdots u_m$ consisting of up to $n-1$ words. Let $G_u(w)$ be that probability for each word w in the vocabulary W. Then we can use a Pitman-Yor process as prior over the distribution $\langle G_u(w) \rangle_{w \in W}$:

$$G_{u_1 \cdots u_m} \sim \mathrm{PY}\left(d_m, \theta_m, G_{u_2 \cdots u_m}\right)$$

where discount d_m and strength θ_m are themselves randomly distributed as we will see below, and the base distribution $G_{u_2 \cdots u_m}$, the vector of probabilities of the current word given all but the earliest word in the context u, is drawn recursively from a Pitman-Yor process in the same manner. This is repeated until we get to G_ε, the distribution of the current word given the empty context, which we assign a prior as follows:

$$G_\varepsilon \sim \mathrm{PY}(d_0, \theta_0, G_0)$$

where the base distribution G_0 is the uniform distribution over W:

$$G_0(w) = 1/|W| \quad \text{for all } w \in W$$

Finally, a uniform prior is placed on the discount parameters d_m and a $\mathrm{Gamma}(1,1)$ prior on the strength parameters θ_m.

Teh (2006) shows that interpolated Kneser-Ney smoothing, a state-of-the-art smoothing method in language modeling, can be interpreted as an approximation of the hierarchical Pitman-Yor model. The paper also describes in detail how to do the inference, using Gibbs sampling.

3.4.2 Experiments and Discussion

To evaluate the language model empirically, perplexity experiments were performed on a 16 million word corpus from APNews. Table 4 shows a subset of the results for interpolated Kneser-Ney

T	n	IKN	MKN	HPY	HPYCV	HDP
2M	3	148.8	144.1	145.7	144.3	191.2
4M	3	137.1	132.7	134.3	132.7	172.7
8M	3	125.9	122.3	123.2	121.9	154.7
14M	3	116.7	113.6	114.3	113.2	140.5
14M	2	169.9	169.2	169.6	169.3	180.6
14M	4	106.1	102.4	103.8	101.9	136.6

Table 4: Perplexities of different n-gram language models for varying n and training corpus sizes T.

(IKN), modified Kneser-Ney (MKN), hierarchical Pitman-Yor (HPY), cross-validation-trained hierarchical Pitman-Yor (HPYCV), and the hierarchical Dirichlet Process (HDP) language models for various sizes of training data. HPY consistently outperforms IKN, which is reassuring given that IKN can be regarded as an approximation of it. MKN's performance is slightly better than HPY. Teh (2006) explains this by the fact that certain parameters in the Kneser-Ney models are optimized using cross-validation, whereas HPY is not optimized towards predictive performance. To validate this claim, he creates an HPY variant HPYCV, in which strength and discount parameters are estimated using cross-validation.[2] The resulting performance is indeed slightly better than MKN.

HDP stands out as extremely badly performing compared to the other models. This should ring alarm bells, as most nonparametric Bayesian models currently used in NLP are based on DPs and HDPs. Whether we buy the power-law argument or not: Since Pitman-Yor includes DP as a special case, we cannot possibly fare worse with it in theory. In practice, however, devising inference algorithms becomes less straight-forward.

4 Conclusion and Future Directions

In this work we gave an overview of several common nonparametric Bayesian models, and reviewed some recent applications of these models to problems in natural language processing. The use of Bayesian priors in general and nonparametric ones in particular is an elegant way of mastering the trade-off between being able to have a powerful model to capture as much detail in the data as possible on the one hand, and having enough evidence in the data to support the model's inferred parameters on the other hand. By providing an explicit generative story for the data relying typically on only a handful of non-random parameters, Bayesian methods are conceptually much cleaner than their maximum-likelihood estimation based alternatives, in which the optimal solution under the model is typically not the best solution of the given task, and where 'meta'-techniques outside of the original model, such as held-out estimation and smoothing, are necessary to produce good practical results.

However, these clean models typically come at the cost of computationally expensive and non-exact inference algorithms. Further, from a pragmatic point of view, none of the applications we reviewed could boast results improving significantly over a smoothed non-Bayesian version of the same model. (Goldwater et al. (2008), who had significant improvements over the best existing segmentation model, did not employ such a baseline model in their experiments, and it could be argued that it would be difficult to come up with a smoothed maximum-likelihood version of their model.)

Nevertheless, the employment of nonparametric Bayesian methods in machine learning and NLP is a field still in its infancy. More insight into what kind of inference algorithms work best in what situation is still needed. Also, new Dirichlet Process variants and generalizations more suitable to specific applications are being found every year.

Indian Buffet Processes To mention just one recent generalization of the Dirichlet Process for which we are not aware of any applications in the NLP community yet, consider the Indian Buffet Process (Griffiths and Ghahramani, 2005). The nonparametric models mentioned so far all enable modeling of data points as belonging to hidden clusters, exactly one cluster per data point. However, consider describing data elements with several latent features. Dirichlet nonparametric models do not support this distributed representation of the data, because cluster membership is exclusive.

Indian Buffet Processes are an extension of Dirichlet Processes that allow a data point to belong to more than one cluster, and be modeled as a set of unbounded latent features. Indian Buffet Processes are thus priors over unbounded binary matrices, with each row being one data point, and each column being one latent feature.

[2]Because of computational limitations, the CV step was based on IKN and therefore approximate.

The application task given by Griffiths and Ghahramani (2005) was for the problem of image reconstruction. We believe that Indian Buffet Process models could be useful to natural language tasks. In part-of-speech tagging based on hidden Markov models, for example, potentially useful features in addition to the standard ones could be the set of words the current word to be tagged co-occurs with in the current sentence: Each element f_v of that feature vector, indexed by the vocabulary V, is one if the current word co-occurs with v and zero otherwise. A natural way of modeling the emission probabilities in a Bayesian fashion would be the Indian Buffet Process.

Further Reading An excellent introduction to Dirichlet Processes is given by Teh (2007). For a deeper treatment of Hierarchical Dirichlet Processes, confer Teh et al. (2006).

The focus of this paper has been on nonparametric models rather than on algorithms to do inference, but for any practical application, finding efficient inference algorithms to estimate the posteriors is crucial. Liang et al. (2007), Goldwater et al. (2008), and DeNero et al. (2008) give detailed descriptions of their variational and/or sampling-based inference algorithms, introduced in a way that can help the reader to adapt them for her own similar nonparametric model. Neal (2000) surveys Markov-Chain Monte Carlo (MCMC) sampling algorithms for inference in Dirichlet Process mixture models. Some MCMC algorithms for inference in HDP mixtures are described by Teh et al. (2006). In a recent paper, Gao and Johnson (2008) compare the quality and efficiency of the Gibbs sampler and variational inference methods, and conclude that for smaller datasets and for the task of HMM sequence labeling, when the number of states are small, Gibbs sampling produces better results at the cost of longer convergence time. Based on their experiments on large datasets, however, variational methods give comparable results to Gibbs sampling with shorter runtime.

References

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

Blackwell, David and James B MacQueen. 1973. Ferguson distributions via Polya urn schemes. *The Annals of Statistics*, 1:353–355.

Blei, David M., Andrew Y. Ng, Michael I. Jordan, and John Lafferty. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:2003.

Brent, Michael R. 1999. An efficient, probabilistically sound algorithm for segmentation and word discovery. *Machine Learning*, 34(1-3):71–105.

Charniak, Eugene and Mark Johnson. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*.

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

DeNero, John and Dan Klein. 2008. The complexity of phrase alignment problems. In *Proceedings of the 46th Annual Meeting on Association for Computational Linguistics (ACL), Short Papers*, pages 25–28, Columbus, Ohio, June. Association for Computational Linguistics.

DeNero, John, Alexandre Bouchard-Côté, and Dan Klein. 2008. Sampling alignment structure under a Bayesian translation model. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Ferguson, Thomas S. 1973. A Bayesian analysis of some nonparametric problems. *The Annals of Statistics*, 1(2):209–230.

Gao, Jianfeng and Mark Johnson. 2008. A comparison of Bayesian estimators for unsupervised Hidden Markov Model POS taggers. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 344–352. Association for Computational Linguistics.

Goldwater, Sharon, Thomas L. Griffiths, and Mark Johnson. 2006. Interpolating between types and tokens by estimating power-law generators. In *Advances in Neural Information Processing Systems (NIPS*, page 18.

Goldwater, Sharon, Thomas L. Griffiths, and Mark Johnson. 2008. A Bayesian framework for word segmentation: Exploring the effects of context. In submission. Available at http://homepages.inf.ed.ac.uk/sgwater/papers/journal-wordseg-hdp.pdf.

Griffiths, Thomas L. and Zoubin Ghahramani. 2005. Infinite latent feature models and the indian buffet process. Technical report, Gatsby Computational Neuroscience Unit Technical Report GCNU TR 2005-001.

Klein, Dan and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics (ACL)*.

Liang, P., S. Petrov, M. I. Jordan, and D. Klein. 2007. The infinite PCFG using hierarchical Dirichlet processes. In *Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP/CoNLL)*.

Mackay, David J. C. and Linda C. Bauman Petoy. 1995. A hierarchical dirichlet language model. *Natural Language Engineering*, 1:1–19.

Matsuzaki, Takuya, Yusuke Miyao, and Jun'ichi Tsujii. 2005. Probabilistic CFG with latent annotations. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics (ACL)*.

Muller, Peter, Fernando Quintana, and Gary Rosner. 2004. A method for combining inference across related nonparametric Bayesian models. *Journal Of The Royal Statistical Society Series B*, 66(3):735–749.

Neal, Radford M. 2000. Markov chain sampling methods for dirichlet process mixture models. *Journal of Computational and Graphical Statistics*, 9(2):249–265.

Pitman, J. and M. Yor. 1997. The two-parameter poisson-dirichlet distribution derived from a stable subordinator. *Annals of Probability*, 25:855–900.

Sethuraman, J. 1994. A constructive definition of Dirichlet priors. *Statistica Sinica*, 4:639–650.

Teh, Y. W., M. I. Jordan, M. J. Beal, and D. M. Blei. 2006. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581.

Teh, Yee Whye. 2006. A hierarchical Bayesian language model based on Pitman-Yor processes. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics (COLING/ACL)*.

Teh, Y. W. 2007. Dirichlet processes. Submitted to Encyclopedia of Machine Learning. Available at www.gatsby.ucl.ac.uk/∼ywteh/research/npbayes/dp.pdf.

Venkataraman, Anand. 2001. A statistical model for word discovery in transcribed speech. *Computational Linguistics*, 27:2001.