Reducing the sampling complexity of topic models

Aaron Q. Li
CMU Language Technologies
Pittsburgh, PA
aaronli@cmu.edu

Amr Ahmed
Google Strategic Technologies
Mountain View, CA
amra@google.com

Sujith Ravi
Google Strategic Technologies
Mountain View, CA
sravi@google.com

Alexander J. Smola
CMU MLD and Google ST
Pittsburgh PA
alex@smola.org

ABSTRACT

Inference in topic models typically involves a sampling step to associate latent variables with observations. Unfortunately, the generative model loses sparsity with the increase in data, requiring $O(k)$ operations per word for k latent states, such as topics. In this paper we propose an algorithm which requires only $O(k_d)$ or less operations per word, where k_d is the number of actually instantiated topics in the document. For large document collections and structured hierarchical models $k_d \ll k$, thus yielding an order of magnitude speedup. Our method is general and it applies to a wide variety of statistical models such as PDP[16, 4] and HDP[19].

At its core is the idea that dense, rapidly changing distributions can be approximated efficiently by the combination of a Metropolis-Hastings step, judicious use of sparsity, and amortized preprocessing via the alias method.

Keywords

Sampling, Scalability, Topic Models

1. INTRODUCTION

Topic models are some of the most versatile tools for modeling statistical dependencies. Given a set of observations $x_i \in \mathcal{X}$, such as documents, logs of user activity, user generated content on microblogs, or communications patterns between participants in an online social network, we want to infer the hidden causes motivating this behavior. A key property in topic models is that they model $p(x)$ via a hidden factor, z via $p(x|z)$ and $p(z)$. More specifically, z is discrete and generally sampling from it is difficult. Examples include where z is the cluster of a document. In this case it leads to Gaussian and Dirichlet mixture models [14]. Whenever z is a vector of topics associated with individual words it leads to Latent Dirichlet Allocation [3]). Likewise, whenever z indicates a term in a hierarchy, it leads to structured and mixed-content annotations [19, 2, 4, 12].

One of the key obstacles in performing scalable inference is to draw $p(z|x)$ from the discrete state distribution associated with the data. A substantial improvement in this context was provided by [22] who decomposed the collapsed sampler [8] for Latent Dirichlet Allocation into components that depend only on the topic sparsity of the document, the topic sparsity of the token, and into a term independent of document and token. As a result the sampling cost can be reduced from $O(k)$, i.e. the total number of topics to $O(k_d + k_w)$, i.e. the number of topics occurring for a particular word w and for a particular document d. This simple modification yielded over an order of magnitude improvement for sampling topic models, thus making their implementation feasible at a large scale. In fact, the strategy is sufficiently robust that it can be extended to settings where the topic smoother depends on the words [15].

For small problems the assumption $k_d + k_w \ll k$ is well satisfied. Unfortunately, as the number of documents grows, one may see immediately that $k_w \to k$, since the probability of observing a particular topic for a given word is rarely nonzero. The following illustrates the problem:

Assume that the probability of occurrence for a given topic for a word is bounded from below by δ. In this case, the probability of the topic occurring at least once is given by

$$1 - (1 - \delta)^n \geq 1 - e^{-n\delta} \to 1 \text{ for } n \to \infty.$$

Moreover, the expected number of topics per word is bounded from below by $k - ke^{-n\delta}$. From this it follows that $k_w = O(k)$ for $n = O(\delta^{-1} \log k)$. In other words, for large numbers of words, the efficiencies discussed in [22] vanish. This is troubling, since in many industrial settings n can be very large, easily in the order of 10^9. In other words, we have more data and the sampler slows down due to the loss of sparsity, thus leading to a superlinear increase in runtime.

On the other hand, the topic-sparsity for a given document essentially remains unchanged, regardless of the total number of related documents that are available. This is due to the fact that the number of tokens per document is typically less than $O(k)$. For instance, microblogs contain only dozens of words, yet admit to thousands of topics. This situation is exacerbated when it comes to hierarchical and structured topic models, since there the number of (sub)topics can grow considerably more rapidly, hence the use of sparsity is crucial in designing efficient samplers.

The present paper proposes a new decomposition of the collapsed conditional probability, in conjunction with a Metropolis-Hastings [7] scheme and the use of the alias method [20, 13] to amortize dense updates for random variables. This method is highly versatile. It defers corrections to the model and avoids renormalization. This allows us to apply it to both flat and hierarchical models. We also provide sampling equations for models using context. Experimental evaluation demonstrates the efficacy of our approach, yielding orders of magnitude acceleration and a simplified algorithm relative to dense problems.

2. TOPIC MODELS

We begin with a brief introduction to topic models and the associated inference problems. This includes a short motivation of sampling schemes in the context collapsed samplers [8, 18] and of stochastic variational models [21]. It is followed by a description of how to extend this to hierarchical models.

2.1 Latent Dirichlet Allocation

In LDA [3] one assumes that documents are mixture distributions of language models associated with individual topics. That is, the documents are generated as follows:

- For each document d draw a topic distribution θ_d from a Dirichlet distribution with concentration parameter α
$$\theta_d \sim \mathrm{Dir}(\alpha).$$
- For each topic t draw a word distribution from a Dirichlet distribution with concentration parameter β
$$\psi_t \sim \mathrm{Dir}(\beta).$$
- For each document draw the number of words, e.g. from a Poisson distribution via
$$n_d \sim \mathrm{Poi}(\gamma).$$
- For each word $i \in \{1 \dots n_d\}$ in document d draw
  - Draw a topic from the multinomial θ_d via
  $$z_{di} \sim \mathrm{Mult}(\theta_d).$$
  - Draw a word from the multinomial $\psi_{z_{di}}$ via
  $$w_{di} \sim \mathrm{Mult}(\psi_{z_{di}}).$$

The beauty of the Dirichlet-multinomial design is that the distributions are conjugate. This means that the multinomial distribution can be integrated out, thus allowing one to express $p(w, z|\alpha, \beta, n_d)$ in a closed-form expression. [8] exploited this to design a Gibbs sampler which samples $p(z_{di}|\text{rest})$ efficiently. Without further ado, the conditional probability is given by

$$p(z_{di}|\text{rest}) \propto \frac{(n_{td}^{-di} + \alpha_t)(n_{tw}^{-di} + \beta_w)}{n_t^{-di} + \bar{\beta}}. \tag{1}$$

Here the count variables n_{td}, n_{tw} and n_t denote the number of occurrences of a particular (topic,document) and (topic,word) pair, or of a particular topic respectively. Moreover, the superscript $-di$ denotes said count when ignoring the pair (z_{di}, w_{di}). For instance, n_{tw}^{-di} denotes the number of times topic t occurs for word w, when ignoring the (topic,word) combination at position (d, i). Finally, $\bar{\beta} := \sum_w \beta_w$ denotes the joint normalization.

At first glance, sampling from (1) appears to cost $O(k)$ time since we have k nonzero terms in a sum that needs to be normalized. [22] devised an ingenious strategy for exploiting sparsity by decomposing terms into

$$p(z_{di}|\text{rest}) \propto \beta_w \frac{\alpha_t}{n_t^{-di} + \bar{\beta}} + n_{td}^{-di} \frac{\beta_w}{n_t^{-di} + \bar{\beta}} + n_{tw}^{-di} \frac{n_{td}^{-di} + \alpha_t}{n_t^{-di} + \bar{\beta}}$$

As can be seen, for small collections of documents only the first term is dense, and more specifically, $\sum_t \alpha_t/(n_t^{-di} + \bar{\beta})$ can be computed from $\sum_t \alpha_t/(n_t + \bar{\beta})$ in $O(1)$ time. That is, whenever both n_{td} and n_{tw} are sparse, sampling from $p(z_{di}|\text{rest})$ can be accomplished efficiently. The use of packed index variables and a clever reordering of (topic,count) pairs further improve efficient sampling to $O(k_w + k_d)$.

Stochastic variational inference [11] requires an analogous sampling step. The main difference being that rather than using $\frac{n_{tw}+\beta_w}{n_t+\bar{\beta}}$ to capture $p(w|t)$ one uses a natural parameter η_{tw} associated with the conjugate variational distribution. Unfortunately this renders the model dense, unless rather careful precautions are undertaken to separate residual dense and sparse components.

In this paper we devise a sampler to draw from $p(z_{di}|\text{rest})$ in amortized $O(k_d)$ time. We accomplish this by building a fast sampler for the following decomposition:

$$p(z_{di}|\text{rest}) \propto n_{td}^{-di} \frac{n_{tw}^{-di} + \beta_w}{n_t^{-di} + \bar{\beta}} + \frac{\alpha_t(n_{tw}^{-di} + \beta_w)}{n_t^{-di} + \bar{\beta}} \tag{2}$$

Here the first term is sparse in k_d. The second term is dense, regardless of the number of documents. In fact, for stochastic variational samplers it is $\alpha_t \eta_{tw}$. However, for both collapsed and stochastic variational methods, the 'language model' $p(w|t)$ does not change too drastically whenever we resample a single word. After all, the number of words is huge, hence the amount of change per word is concomitantly small. We exploit this property by building a sampler that is able to draw in $O(k_d)$ time from an approximation of (2).

2.2 Poisson Dirichlet Process

Before going into details of our algorithm for 'flat' topic models, we briefly give an overview of hierarchical models. They share many aspects with flat models when it comes to sampling. More specifically, they also decompose into a component that largely depends on the sparsity pattern with a given document and a component dependent on the sparsity (or lack thereof) of the generative process.

In a typical hierarchical Bayesian topic model, the topic-word multinomial distribution is often sampled from a parametrised distribution family $F(\tau, \psi_o)$, where τ is the set of parameters and ψ_0 is a generated base distribution. In Pitman-Yor Topic Model (PYTM) [17], the chosen distribution family is the three-parameter Poisson Dirichlet Process (a.k.a Pitman-Yor Process), with a base distribution generated by $Dirichlet(\vec{\beta})$. In [5] it has also been shown that the mixture provides a systematic alternative to models in computer vision that first cluster image patches and then compute e.g. topic models on the computed 'visual words'.

The Poisson Dirichlet Process $PDP(b, a, \psi_0)$ [16, 4] captures the power-law property of natural language corpus as observed in Zipf's law by allowing generated distributions to randomly share a countable subset of atoms that is con-

tained in the base distribution ψ_0 with different weightings. With respect to samples drawn from a distribution generated from $PDP(b, a, \psi_0)$, the discount parameter a penalizes chance of frequent samples to be sampled again from a generated distribution, and the concentration parameter a encourages the set of samples that already been drawn to appear more often, which can be observed as a clustering effect, and long tail distribution of word occurrences when applied to nature languages.

We use the following variant of the PYTM, known as the Differential Topic Model [5], with a more effective inferencing scheme from [6]. This variant has much simpler structure than the Hierarchical Pitman-Yor Topic Model in [17], applies power-law property to both documents and topics, and excels in the performance when applied to differential analysis for multiple groups of documents. For the ease of comparing the results with other models, we reduce the model to a simple form by assuming only one group of document is present, and no word-correlation transformation matrix is applied to base distribution of Poisson-Dirichlet Process:

$$\begin{aligned}
\vec{\psi}_t^0 &\sim Dirichlet(\vec{\beta}) && \forall t = 1, ..., k \\
\vec{\psi}_t &\sim PDP(b_t, a_t, \vec{\psi}_t^0) && \forall t = 1, ..., k \\
\vec{\theta}_d &\sim Dirichlet(\vec{\alpha}) && \forall d = 1, ..., D \\
z_{di} &\sim Discrete(\vec{\theta}_d) && \forall d = 1, ..., D, \forall i = 1, ..., n_d \\
w_{di} &\sim Discrete(\vec{\psi}_{z_{di}}) && \forall d = 1, ..., D, \forall i = 1, ..., n_d
\end{aligned}$$

. Where b_t and a_t are the concentration and discount parameters respectively for each topic t. To perform inference in this model, we follow the same steps given in [6] and [5], and consider the Poisson Dirichlet Process as an equivalent Chinese Restaurant Process, where word types are considered as dishes served to tables of customers, and words in each documents are customers sitting in these tables. i.e. each topic is treated as a restaurant. Note here that the hierarchy is over word-topics but the model is still parametric in the number of topics. We introduce auxiliary variables $s_{t,w}$, the number of tables served with dish w in restaurant t (topic t), $r_{d,i}$ represents whether the i-th word in document d has caused the creation of a new table, and $m_{t,w}$, the number of dishes that have been served in restaurant t (topic t), a variable similar to how n_{wk} functions in LDA.

The conditional probability is given by:

$$\begin{aligned}
&p(z_{d,i} = t, r_{d,i} = 0| \text{ the rest }) \qquad (3) \\
&\propto \frac{\alpha_t + n_{dt}}{b_t + m_t} \frac{m_{tw} - s_{tw} + 1}{m_{tw} + 1} \frac{S^{m_{tw}+1}_{s_{tw},a_t}}{S^{m_{tw}}_{s_{tw},a_t}}
\end{aligned}$$

$$\begin{aligned}
&p(z_{d,i} = t, r_{d,i} = 1| \text{ the rest}) \qquad (4) \\
&\propto (\alpha_t + n_{dt}) \frac{b_t + a_t s_t}{b_t + m_t} \frac{s_{tw} + 1}{m_{tw} + 1} \frac{\gamma + s_{tw}}{\bar{\gamma} + s_t} \frac{S^{m_{tw}+1}_{s_{tw}+1,a_t}}{S^{m_{tw}}_{s_{tw},a_t}}
\end{aligned}$$

Where $S^N_{M,a}$ is the generalised Stirling number[1], a quantity which is readily tabulated . $m_t = \sum_w m_{tw}$ and $s_t = \sum_t s_{tw}$. Using the same strategy we used with LDA, the above two expressions can be broken down into a sparse term plus a dense term.

$$\begin{aligned}
&p(z_{d,i} = t, r_{d,i} = 0| \text{ the rest }) \qquad (5) \\
&\propto \frac{n_{dt}}{b_t + m_t} \frac{m_{tw} - s_{tw} + 1}{m_{tw} + 1} \frac{S^{m_{tw}+1}_{s_{tw},a_t}}{S^{m_{tw}}_{s_{tw},a_t}} \\
&\quad + \frac{\alpha_k}{b_t + m_t} \frac{m_{tw} - s_{tw} + 1}{m_{tw} + 1} \frac{S^{m_{tw}+1}_{s_{tw},a_t}}{S^{m_{tw}}_{s_{tw},a_t}}
\end{aligned}$$

$$\begin{aligned}
&p(z_{d,i} = t, r_{d,i} = 1| \text{ the rest}) \qquad (6) \\
&\propto n_{dt} \frac{b_t + a_t s_{tw}}{b_t + m_{tw}} \frac{s_{tw} + 1}{m_{tw} + 1} \frac{\gamma + s_{tw}}{\bar{\gamma} + s_t} \frac{S^{m_{tw}+1}_{s_{tw}+1,a_t}}{S^{m_{tw}}_{s_{tw},a_t}} \\
&\quad + \alpha_t \frac{b_t + a_t s_{tw}}{b_t + m_t} \frac{s_{tw} + 1}{m_{tw} + 1} \frac{\gamma + s_{tw}}{\bar{\gamma} + s_t} \frac{S^{m_{tw}+1}_{s_{tw}+1,a_t}}{S^{m_{tw}}_{s_{tw},a_t}}
\end{aligned}$$

2.3 Hierarchical Dirichlet Process

To illustrate the efficacy of our approach we discuss a third case, namely the setting where we have a hierarchical mixture on topic emission. In contrary to the previous model where the hierarchical structure is applied to the emission of the topic-word distribution, the topic models in this category assume a hierarchical mixture exist on the generation of topic or document-topic distributions.

A typical well studied topic model with such property is the LDA topic model mixed in with Hierarchical Dirichlet Process [19] (HDP-LDA). The Dirichlet Process, denoted by $DP(b, H(\cdot))$, is merely a special case of the Poisson Dirichlet Process $PDP(b, a, H(\cdot))$ where the discount parameter is set to 0. In the settings of this topic model the topic mixture $\vec{\theta}_d^1$ for each document d is drawn from a Dirichlet Process $DP(b_1, \theta_d^0)$, where the base distributions θ_d^0 for all documents are generated by a base Dirichlet Process $DP(b_0, H)$, thus allowing each document to have its own individual mixture of topic representations, while at the same time enabling topic components to be shared across all documents through a latent common base mixture. The complete generative process is as the following:

$$\begin{aligned}
\vec{\psi}_t &\sim Dirichlet(\vec{\gamma}) && \forall t = 1, ..., k \\
\vec{\theta}_d^0 &\sim DP(b_0, H(.)) && \forall d = 1, ..., D \\
\vec{\theta}_d^1 &\sim DP(b_1, \theta_d^0) && \forall d = 1, ..., D \qquad (7) \\
z_{di} &\sim Discrete(\vec{\theta}_d^1) && \forall d = 1, ..., D, \forall i = 1, ..., n_d \\
w_{di} &\sim Discrete(\vec{\psi}_{z_{di}}) && \forall d = 1, ..., D_i, \forall i = 1, ..., n_d
\end{aligned}$$

. where $\vec{\psi}_t$ is the topic-word distribution with $\vec{\gamma}$ prior. $H(\cdot)$ is the base distribution over topics for the base DP.

The Hierarchical Dirichlet Process allows the number of layers of Dirichlet Process to be freely extended beyond two levels. These extensions, including an infinite mixture model [19], enables a richer representation of the hierarchical structure in topics. An equivalent Chinese Restaurant Franchise analogy [6, 19] is often given by mapping each Dirichlet Process to a single Chinese Restaurant Process, as well as the hierarchical structure identifying each restaurant's parents and children. Let N_j be the total number of customers in

[1]The generalised Stirling number $S^N_{M,a}$ is given by the recursion $S^{N+1}_{M,a} = S^N_{M-1,a} + (N - Ma)S^N_{M,a}$, $S^N_{M,a} = 0$ for $M > N$, and $S^N_{0,a} = \delta_{N,0}$. See [4] for more details.

restaurant j, and $n_{j,t}$ be the number of customers in restaurant j having dish t. When a new customer (a word) enters a restaurant j, denoted by $DP(b_j, H_j(\cdot))$, with probability $\forall t : \frac{n_{j,t}}{b_j+N_j}$ the customer is served with dish t (topic t) and sits in an existing table, and with probability $\frac{b_j}{b_j+N_j}$ the customer sits in a new table serving dish t drawn from $H_j(\cdot)$. In the event that the customer sits in a new table, a proxy customer would be sent to the parent restaurant of j, denoted by j' and $DP(b_{j'}, H_{j'}(\cdot))$. The proxy customer then decide whether a new table is chosen or not in restaurant j' with respect to the probability between $\forall t : \frac{n_{j',t}}{b_{j'}+N_{j'}}$ and $\frac{b_{j'}}{b_{j'}+N_{j'}}$. This process repeats and only terminates when there is no more parent restaurant or the customer decide to sit in an existing table in any restaurant along the path.

To demonstrate our strategy is applicable to the general multilevel Hierarchical Dirichlet Process, we use the block Gibbs sampler given in [6] as the basis for extension, as probabilistically block Gibbs sampler is generally preferable for better mixing and faster convergence, and empirical evidence shows a better performance than the Sampling by Direct Assignment method given in [19] and the Collapsed Gibbs Table Sampler given in [4]. The conditional probability for the two-level HDP-LDA (equation 7) is given by:

$$p(z_{di} = t,\, u_{di} = u|\text{the rest}) \tag{8}$$
$$\propto \begin{cases} \frac{b_0 b_1}{b_0+s}\frac{\gamma_w+m_{tw}}{\bar{\gamma}+m_t} & \text{if } s_{t0} = 0 \\ \frac{S^{n_{dt}+1}_{s_{dt}+1,0}}{S^{n_{dt}}_{s_{dt},0}}\frac{s_{dt}+1}{n_{dk}+1}\frac{\gamma_w+m_{tw}}{\bar{\gamma}+m_t} & \text{if } s_{td} \neq 0,\ s_{t0} \neq 0 \\ \frac{b_1 s_t^2}{(s_t+1)(s+b_0)}\frac{\gamma_w+m_{tw}}{\bar{\gamma}+m_t} & \text{if } s_{td} = 0,\ s_{t0} \neq 0 \end{cases}$$

Where u is a indicator variable showing up to which level the word w has contributed a table (in the two level settings it can only take the value 0 and 1. See [6] for a detailed explanation on the restraints), s_{t0} is the table counter for the base Dirichlet Process, $s = \sum_t s_t$, and the rest of the terms carry the same meaning as defined in the last section.

A generalised form is given as :

$$p(z_l = t,\, u_l = u|\text{the rest}) \tag{9}$$
$$\propto \begin{cases} \frac{b_0 b_1}{b_0+s}\frac{\gamma_w+m_{tw}}{\bar{\gamma}+m_t} & \text{if } s_{t0} = 0 \\ \frac{S^{''n_{dt}}_{s''_{dt},0}}{S^{'n_{dt}}_{s'_{dt},0}}\frac{\gamma_w+m_{tw}}{\bar{\gamma}+m_t}\frac{(s''_{dt})^{\delta_{t''_{dt}\neq t'_{dt}}}}{(n''_{dt})^{\delta_{n''_{dt}\neq n'_{dt}}}}. & \text{(...continuing)} \\ (n''_{dt} - t''_{dt})^{\delta_{n''_{td}-t''_{td}\neq n'_{td}-t'_{td}}} & \text{if } s_{td} \neq 0,\ s_{t0} \neq 0 \\ \frac{b_1 s_t^2}{(s_t+1)(s+b_0)}\frac{\gamma_w+m_{tw}}{\bar{\gamma}+m_t} & \text{if } s_{td} = 0,\ s_{t0} \neq 0 \end{cases}$$

Where l is referring to a non-proxy customer in any restaurant. x' refer to the variable value before removing l, and x'' is referring to its value after adding back l, according to the constraints given in [6].

In both the HDP-LDA form and the generalised form, one can immediately see that the term m_{tw} is sparse. Therefore, both equation 8 and 9 can be decomposed to a dense term multiplied by γ_w, and a sparse term multiplied by m_{tw}. Applying the same methodology, it is not hard to see we can reduce the sampling complexity of a general Hierarchical Dirichlet Process with a constant number of layers to $O(k_w)$.

3. A FAST SAMPLER

We now introduce the key components for our FAST Algorithm to Sample Topics. They consist of the alias method [20, 13] and a simplified version of the Metropolis-Hastings sampler [7].

3.1 The Alias Method

Typically, when drawing from a distribution over l outcomes, it is accepted that one would need to perform $O(l)$ work to generate a sample. In fact, this is a lower bound, since we need to inspect each probability at least once before we can construct the sampler. However, what is commonly overlooked is that there exist algorithms that allow us to draw *subsequent samples from the same distribution* in $O(1)$ time. This means that drawing l samples from a distribution over l outcomes can be accomplished in $O(1)$ amortized time per draw.

The algorithm works by decomposing a distribution over l events into l bins of equal probability by pairing at most two events per bin. Since it 'robs' from the probabilities $p_i > 1/l$ and adds to those with $p_i < 1/l$ it is also referred to as 'Robin Hood' method [13].

Denote by p_i with $i \in \{1 \dots l\}$ the probabilities from which we would like to draw. The algorithm proceeds as follows:

GenerateAlias(p, l)

Set lists $L = H = \emptyset$ and array $A = []$.
for $i = 1$ **to** l **do**
  if $p_i \leq l^{-1}$ **then** $L \leftarrow L \cup \{(i, p_i)\}$ **else** $H \leftarrow H \cup \{(i, p_i)\}$
end for
while $L \neq \emptyset$ **do**
  Extract (i, p_i) from L and (h, p_h) from H
  $A \leftarrow [A, (i, h, p_i)]$
  if $p_h - p_i > l^{-1}$ **then**
    $H \leftarrow H \cup \{(h, p_h - p_i)\}$
  else
    $L \leftarrow L \cup \{(h, p_h - p_i)\}$
  end if
end while
return A

SampleAlias(A, l)

$(a, b, \pi) = A[\text{RandInt}(l)]$
if $l\pi >$ RandUnif(1) **then return** a **else return** b

We first build the Alias table (A) ONCE using GenerateAlias where each entry is a triplet (that algoritm takes $O(l)$ time and space). To draw a sample, the algorithm works by first drawing one of the l bins stored in A uniformly at random (using RandInt(l)). This is achieved in $O(1)$ time for $l < 2^{64}$ given 64 bit microarchitectures. Subsequently we flip a biased coin based on the probability stored with the triplet (l_π) to decide between outcomes a (from probability table) and b (from alias table). A decomposition of p into A can be proven by recursion (see [13] for more details).

3.2 Approximate Alias Sampling

Whenever we draw l identical samples from p it is clear that the above algorithm provides an $O(1)$ sampler. However, if p changes, it is difficult to apply the alias sampler directly. To address this, we use rejection sampling and Metropolis-Hastings procedures. Rejection sampling proceeds as follows:

Rejection(p, q, c)

repeat
Draw $i \sim q(i)$
until $p(i) \geq cq(i)\text{RandUnif}(1)$
return i

Here p is the distribution we would like to draw from, q is a reference distribution that makes sampling easy, and $c \geq 1$ is chosen such that $cq(i) \geq p(i)$ for all i. We then accept with probability $\frac{p(i)}{cq(i)}$. It is well known that the expected number of samples to draw via **Rejection**(p, q, c) is c, provided that a good bound c exists. In this case we have the following lemma:

Lemma 1 *Given l distributions p_i and q over l outcomes satisfying $c_i q \geq p_i$, the expected amortized runtime complexity for drawing using **SampleAlias**(A, l) and rejecting using **Rejection**(p_i, q, c_i) is given by $O\left(\frac{1}{l}\sum_{i=1}^{l} c_i\right)$.*

Proof. Preprocessing costs amortized $O(1)$ time. Each rejection sampler costs $O(c_i)$ work. Averaging over the draws proves the claim. □

In many cases, unfortunately, we do not know c_i, or computing c_i is essentially as costly as drawing from p_i itself. Moreover, in some cases c_i may be unreasonably large. In this situation we resort to Metropolis Hastings sampling using a stationary proposal distribution. That is, given a state i and using a transition probability $q(j|i)$ for a move from $i \to j$ we accept the new state with probability $\min\left(1, \frac{p(j)q(i|j)}{p(i)q(j|i)}\right)$. The key difference is that now the set of states forms a *chain* where the elements are *not independent* of each other.

Good sampling is achieved whenever the chain mixes rapidly. The advantage in the context of statistical inference is that we do not start from an unsuitable state. Moreover, we can assume that the state is unlikely to change substantially once the model starts to converge, since the state is associated with the cluster ID, or the topic ID of a word, image patch, or any similar object. We use a particularly simple variant of the above scheme, namely $q(j|i) = q(i)$. That is, we assume that the proposal is independent of the current state. This simplifies the acceptance probability to $\min\left(1, \frac{p(j)q(i)}{p(i)q(j)}\right)$. We obtain the following:

StationaryMetropolisHastings(A, l, p, n)
if no initial state exists **then** $i =$ **SampleAlias**(A, l)
for $i = 1$ **to** n **do**
Draw $j =$ **SampleAlias**(A, l)
if $\text{RandUnif}(1) < \min\left(1, \frac{p(j)q(i)}{p(i)q(j)}\right)$ **then** $i \leftarrow j$
end for
return i

Lemma 2 *If the Metropolis Hastings sampler using q instead of p mixes sufficiently well in n steps, the amortized cost of drawing n samples from q is $O(1)$ per sample.*

This follows directly from the construction of the sampler and the fact that we can amortize generating the alias table. An obvious requirement for the above algorithm to work is that $\frac{q(i)}{q(j)}$ is well-defined throughout. A sufficient condition for this is that $q(j) > 0$ for all j. We will exploit this later.

3.3 Sampler for LDA

We now have all components that are required for an accelerated sampler. The trick is to recycle old values for $p(w_{di}|z_{di})$ even when they change slightly and then to correct this via a Metropolis-Hastings scheme. Since the values change only slightly, we can therefore amortize the values efficiently. We begin by discussing this for the case of 'flat' topic models and extend it to hierarchical models subsequently.

Recall (2). We now design a suitable proposal distribution for it. It involves computing the document-specific sparse term exactly and approximating the remainder with slightly stale data. Furthermore, to avoid the need to store a stale alias table, we simply *draw* from the distribution and keep a supply of samples. Once this is used, at which point we have amortized generating the table, we simply recompute a new table. We begin by describing the algorithm:

Alias table: Denote by

$$Q_w := \sum_t \alpha_t \frac{n_{tw} + \beta_w}{n_t + \bar{\beta}} \text{ and } q_w(t) := \frac{\alpha_t}{Q_w} \frac{n_{tw} + \beta_w}{n_t + \bar{\beta}}$$

the alias normalization and the associated probability distribution. Then we perform the following steps:

1. Generate the alias table A using q_w.
2. Draw k samples from q_w and store them in S_w.
3. Discard A and only retain Q_w and the array S_w.

Generating S_w and computing Q_w costs $O(k)$ time. In particular, storage of S_w requires at most $O(k \log_2 k)$ bits, thus it is much more compact to store than A. Note, however, that we need to store $q_w(t)$ and the old snapshot of n_{tw} and n_t for the purpose of our sampler. In the following we denote these quantities with a bar, i.e. $\bar{n}_{tw}$ and $\bar{n}_t$.

Metropolis Hastings proposal: Denote by

$$P_{dw} := \sum_t n_{td}^{-di} \frac{n_{tw}^{-di} + \beta_w}{n_t^{-di} + \bar{\beta}} \text{ and } p_{dw}(t) := \frac{n_{td}^{-di}}{P_{dw}} \frac{n_{tw}^{-di} + \beta_w}{n_t^{-di} + \bar{\beta}}$$

the sparse document-dependent topic contribution. Computing it costs $O(k_d)$ time. This allows us to construct a proposal distribution

$$q(t) := \frac{P_{dw}}{P_{dw} + Q_w} p_{dw}(t) + \frac{Q_w}{P_{dw} + Q_w} q_w(t) \tag{10}$$

To perform an MH-step we then draw from $q(t)$ in $O(k_d)$ amortized time. The step from topic s to topic t is accepted with probability $\min(1, \pi)$ where

$$\pi = \frac{n_{td}^{-di} + \alpha_t}{n_{sd}^{-di} + \alpha_s} \cdot \frac{n_{tw}^{-di} + \beta_w}{n_{sw}^{-di} + \beta_w} \cdot \frac{n_s^{-di} + \bar{\beta}}{n_t^{-di} + \bar{\beta}} \cdot \frac{P_{dw} p_{dw}(s) + Q_w q_w(s)}{P_{dw} p_{dw}(t) + Q_w q_w(t)}$$

Note that the last fraction effectively removes the normalization in p_{dw} and q_w respectively, that is, we take ratios of unnormalized probabilities.

Complexity: To draw from q costs $O(k_d)$ time. This is so since computing P_{dw} has this time complexity, and so does the sampler for p_{dw}. Moreover, drawing from $q_w(t)$ is $O(1)$, hence it does not change the order of the algorithm. Note that repeated draws from q are only $O(1)$ since we can use the very same alias sampler also for draws from p_{dw}. Finally, evaluating π costs only $O(1)$ time. We have the following:

Lemma 3 (FAST Sampler) *Drawing up to k steps in a Metropolis-Hastings proposal from $p(z_{di}|\text{rest})$ can be accomplished in $O(k_d)$ amortized time and $O(k)$ space.*

3.4 Sampler for Poisson Dirichlet Process

The same procedure we use to derive the sampler for LDA can be applied for the basic Poisson Dirichlet Process Hierarchical model by utilising the sparsity of n_{dt}, with a slight tuning to merge variables t and r into a single variable v. Define $v = 1, ..., 2k$, and let the value of v denote a sample pair $(t, r) = (\lfloor \frac{v}{2} \rfloor,\ \text{v mod } 2)$ (recall that r is a binary variable denoting if word w introduces a new table in topic t). Recall equation 5, The alias table is now denoted by:

$$\begin{aligned} Q_w : &= \sum_v [(v+1 \bmod 2)(\frac{\alpha_k}{b_t + m_{tw}} \frac{m_{tw} - s_{tw} + 1}{m_{tw} + 1} \frac{S^{m_{tw}+1}_{s_{tw},a_t}}{S^{m_{tw}}_{s_{tw},a_t}}) \\ &\quad + (v \bmod 2)\alpha_k \frac{b_t + a_t s_t}{b_t + m_t} \frac{s_{tw} + 1}{m_{tw} + 1} \frac{\gamma + s_{tw}}{\bar{\gamma} + s_t} \frac{S^{m_{tw}+1}_{s_{tw}+1,a_t}}{S^{m_{tw}}_{s_{tw},a_t}}] \end{aligned}$$

and

$$q_w(v) = \begin{cases} \frac{\alpha_k}{b_t + m_{tw}} \frac{m_{tw} - s_{tw} + 1}{m_{tw} + 1} \frac{S^{m_{tw}+1}_{s_{tw},a_t}}{S^{m_{tw}}_{s_{tw},a_t}} & \text{if v\%2=0} \\ \alpha_k \frac{b_t + a_t s_t}{b_t + m_t} \frac{s_{tw} + 1}{m_{tw} + 1} \frac{\gamma + s_{tw}}{\bar{\gamma} + s_t} \frac{S^{m_{tw}+1}_{s_{tw}+1,a_t}}{S^{m_{tw}}_{s_{tw},a_t}} & \text{otherwise} \end{cases}$$

Metropolis Hastings proposal: Denoted by

$$\begin{aligned} P_{dw} : &= \sum_v n_{d,\lfloor \frac{v}{2} \rfloor} [\\ &\quad (v \bmod 2) \frac{1}{b_t + m_t} \frac{m_{tw} - s_{tw} + 1}{m_{tw} + 1} \frac{S^{m_{tw}+1}_{s_{tw},a_t}}{S^{m_{tw}}_{s_{tw},a_t}} \\ &\quad + (v+1 \bmod 2) \frac{b_t + a_t s_{tw}}{b_t + m_{tw}} \frac{s_{tw} + 1}{m_{tw} + 1} \frac{\gamma + s_{tw}}{\bar{\gamma} + s_t} \frac{S^{m_{tw}+1}_{s_{tw}+1,a_t}}{S^{m_{tw}}_{s_{tw},a_t}}] \end{aligned}$$

and

$$p_{dw}(v) : = \begin{cases} \frac{n_{d,\lfloor \frac{v}{2} \rfloor}}{b_t + m_t} \frac{m_{tw} - s_{tw} + 1}{m_{tw} + 1} \frac{S^{m_{tw}+1}_{s_{tw},a_t}}{S^{m_{tw}}_{s_{tw},a_t}} & \text{if v\%2=0} \\ \frac{n_{d,\lfloor \frac{v}{2} \rfloor}(b_t + a_t s_{tw})}{b_t + m_{tw}} \frac{s_{tw} + 1}{m_{tw} + 1} \frac{\gamma + s_{tw}}{\bar{\gamma} + s_t} \frac{S^{m_{tw}+1}_{s_{tw}+1,a_t}}{S^{m_{tw}}_{s_{tw},a_t}} & \text{otherwise} \end{cases}$$

Similarly, computing $p_{dw}(v)$ only costs $O(k_d)$ time, which allows us to construct a similar proposal distribution:

$$q(v) : = \frac{P_{dw}}{P_{dw} + Q_w} p_{dw}(v) + \frac{Q_w}{P_{dw} + Q_w} q_w(v)$$

And acceptance probability $\min(1, \pi)$ of the MH-step of the transposition $v_1 \to v_2$, where $v_1 = (t_1, r_1)$, $v_2 = (t_2, r_2)$, and :

$$\begin{aligned} \pi &= \frac{p(z_{d,i} = t_2, r_{d,i} = r_2 | \text{ the rest }) q(v_1)}{p(z_{d,i} = t_1, r_{d,i} = r_1 | \text{ the rest }) q(v_2)} \\ &= \frac{p(z_{d,i} = t_2, r_{d,i} = r_2 | \text{ the rest })}{p(z_{d,i} = t_1, r_{d,i} = r_1 | \text{ the rest })} \frac{P_{dw} p_{dw}(v_1) + Q_w q_w(v_1)}{P_{dw} p_{dw}(v_2) + Q_w q_w(v_2)} \end{aligned}$$

Where $p(...|\text{ the rest})$ is given in equation 3 and 4. Following the same argument we made for the LDA sampler, it can be seen that the complexity of this sampler is also $O(k_d)$ in amortised time.

3.5 Sampler for Hierarchical Dirichlet Process

The procedure is exactly the same except we now utilise the sparsity on word-topic counts rather than document-topic counts. This is beneficial in the cases when large amount of low frequency words exists.

For brevity we only show the inference steps for the two level case (HDP-LDA) but the general case can be easily extended from that. Define $v = 1, ..., 2k$, and let the value of v denote a sample pair $(t, u) = (\lfloor \frac{v}{2} \rfloor,\ \text{v mod } 2)$ (recall that u can take values $\{0, 1\}$ in two-level models). given equation 8, the alias table is now denoted by:

$$Q_w : = \gamma_w \sum_v p(z_{di} = \frac{v}{2}, u_{di} = v \bmod 2)(\bar{\gamma} + m_{\frac{v}{2}})$$

and

$$q_w(v) = p(z_{di} = \frac{v}{2}, u_{di} = v \bmod 2)(\bar{\gamma} + m_{\frac{v}{2}})\gamma_w$$

Metropolis Hastings proposal: Denoted by

$$P_{dw} : = \gamma_w \sum_v p(z_{di} = \frac{v}{2}, u_{di} = v \bmod 2)(\bar{\gamma} + m_{\frac{v}{2}}) m_{\frac{v}{2}w}$$

and

$$p_{dw}(v) : = \gamma_w p(z_{di} = \frac{v}{2}, u_{di} = v \bmod 2)(\bar{\gamma} + m_{\frac{v}{2}}) m_{\frac{v}{2}w}$$

With the same proposal distribution:

$$q(v) : = \frac{P_{dw}}{P_{dw} + Q_w} p_{dw}(v) + \frac{Q_w}{P_{dw} + Q_w} q_w(v)$$

And acceptance probability $\min(1, \pi)$ for the transtition $v_1 \to v_2$, where $v_1 = (t_1, u_1)$, $v_2 = (t_2, u_2)$:

$$\begin{aligned} \pi &= \frac{p(z_{d,i} = t_2, u_{d,i} = u_2 | \text{ the rest }) q(v_1)}{p(z_{d,i} = t_1, r_{d,i} = r_1 | \text{ the rest }) q(v_2)} \\ &= \frac{p(z_{d,i} = t_2, u_{d,i} = u_2 | \text{ the rest })}{p(z_{d,i} = u_1, r_{d,i} = u_1 | \text{ the rest })} \frac{P_{dw} p_{dw}(v_1) + Q_w q_w(v_1)}{P_{dw} p_{dw}(v_2) + Q_w q_w(v_2)} \end{aligned}$$

where $w_{di} = w$ is omitted for brievity, the same argument can show that the time complexity of our sampler for HDP-LDA is amortised $O(k_w)$.

4. EXPERIMENTS

To show the empirical performance of the alias method we implemented the aforementioned samplers in both its base forms that has $O(k)$ time complexity, as well as our alias variants which have amortised $O(k_w)$ or $O(k_d)$ time complexity. In addition to this, we implemented the SparseLDA [22] algorithm with the full set of features including the sorted list containing compact encoding of n_{tw} and n_{dt}, as well as dynamic $O(1)$ update of bucket values. Beyond the standard implementation provided in MalletLDA by [22], we made two major improvements: 1) the optimisation of the speed of the sorting algorithm for the compact list of encoded values to amortised $O(1)$; and 2) avoiding the use of hash maps therefore substantially improved the speed in general with small sacrifice of memory efficiency by using inverted list on the indices (and inverted list of the indices of the inverted lists) of the counts and auxiliary variables.

In this section these implementations will be referred as **LDA** ($O(k)$), **SparseLDA**($O(k_w + k_d)$), **AliasLDA**($O(k)$), **PDP** ($O(k)$) [5], **AliasPDP** ($O(k_d)$), **HDP** ($O(k)$) [6], and **AliasHDP** ($O(k_w)$).

4.1 Environment and Datasets

All our implementations are written in C++11 in a way that maximise runtime speed, compiled with g++4.8 with -O3 compiler optimisation in amd64 architecture. All our experiments are conducted on a laptop machine 12GB memory, and an Intel i7-740QM processor with only 1.73GHz clock rate, 4×256KB L2 Cache and 6MB L3 Cache. Furthermore, we only use one single sampling thread across all experiments. Therefore, only one CPU core is active throughout an entire experiment, and only 256KB L2 cache is available. We further disabled Turbo Boost to ensure all experiment are run at exactly 1.73GHz clock rate. We use Ubuntu 13.10 64bit as the runtime environment.

We use 5 datasets with a variety in sizes, vocabulary length, and document lengths for evaluation, as shown in Table 1 . **RedState** dataset contains American political blogs crawled from redstate.com in year 2011. **GPOL** contains a subset of political news articles from Reuters RCV1 collection [2]. **Enron** is the well known Enron Email Dataset [3]. **NYTimes** contains articles published by New York Times between year 1987 and 2007. **PubMedSmall** is a small subset (around 1%) of the biomedical literature abstracts **PUBMED**. Stopwords are removed from all datasets. Furthermore, words occurring less than 10 times are removed from **NYTimes**, **Enron**, and **PubMedSmall** datasets. **NYTimes**, **Enron**, and **PUBMED** datasets are available at [1] .

4.2 Evaluation Metrics and Parameter Settings

We evaluate the algorithms based on two metrics: amount of time elapsed for one Gibbs sampling iteration and perplexity. The perplexity is evaluated for every 5 iterations, beginning with the first evaluation at the ending of the first Gibbs sampling iteration. We use the standard held-out method [9] to evaluate test perplexity, in which a small set of test document originated from the same collection is set to query the model being trained and produces an estimate of the document-topic mixtures $\tilde{\theta}_{dt}$ for each test document d. From there the perplexity is then evaluated as:

$$P(\mathbf{W}|\text{ the rest}) = \prod_{d=1}^{D} p(\vec{w}_d|\text{ the rest})^{-\frac{1}{N}}$$
$$= exp(-\frac{\sum_{d=1}^{D} log(p(\vec{w}_d|\text{ the rest}))}{\sum_{d=1}^{D} N_d})$$

And

$$p(\vec{w}_d|\text{ the rest}) = \prod_{i=1}^{n_d}\sum_{t=1}^{k} p(w_i = w|z_{d,i} = t)p(z_{d,i} = t)$$
$$= \prod_{i=1}^{n_d}\sum_{t=1}^{k} (\psi_{tw}\tilde{\theta}_{dt})^{n_{dt}}$$

[2]Reuters Corpus, Volume 1, English language, 1996-08-20 to 1997-08-19 (Release date 2000-11-03).
[3]Original source: www.cs.cmu.edu/~enron

Dataset	V	L	D	T	L/V	L/D
RedState	12,272	321,699	2,045	231	26.21	157
GPOL	73,444	2,638,750	14,377	1,596	35.9	183
Enron	28,099	6,125,138	36,999	2,860	218	165
PubMedSmall	106,797	35,980,539	546,665	2,002	337	66
NYTimes	101,636	98,607,383	297,253	2,497	970	331

Table 1: Datasets and their statistics. V Denotes vocabulary size, L denotes total number of training tokens, D denotes total number of training documents. T denotes total number of test documents. L/V indicates the average number occurrences of a word. L/D indicates the average document length.

Where an estimate of ψ_{tw} is obtained from the model being trained.

We set the number of topics $k = 1024$ for all experiments except one (GPOL) where we vary k and observe the effect on speed per iteration. We use fixed values for hyperparameters in all our models. The parameters are given as $\alpha = \beta = 0.1$ for LDA, $a = 0.1$, $b = 10$, and $\gamma = 0.1$ for PDP, and $b_0 = b_1 = 10$, $\gamma = 0.1$ for HDP. For alias implementations, we fix the number of Metropolis-Hasting sampling steps at 2, as we observed a satisfactory acceptance rate (>90%) at these settings, and only as negligible improvement in perplexity by raising this value. Furthermore, we did not observe degraded topic quality even when Metropolis-Hasting sampling step is reduced to 1, and in all our experiments the perplexity almost perfectly converges at the same pace (i.e. along number of iterations) with the same algorithm without applying alias method (but with a much faster iteration time).

4.3 Results

4.3.1 *Overall Performances*

Figure 1: Perplexity over running time for LDA, SparseLDA, and AliasLDA

Figure 1 shows the overall performance of perplexity v.s time elapsed comparing SparseLDA vs AliasLDA on the four larger datasets. When k is fixed to 1024, substantial performance in perplexity over running time (Figure 1) is gained except for Enron dataset, most likely due to its uniquely small vocabulary size. The gap in performance is increased

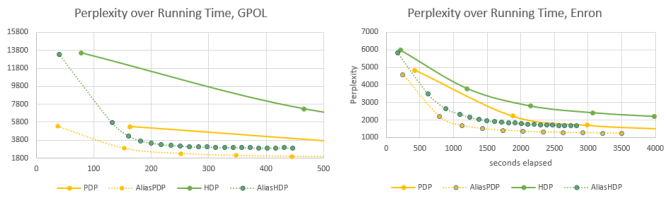

Figure 2: Perplexity over running time for PDP, AliasPDP, HDP, and AliasHDP

as the datasets become larger and more realistic in size. The gain in performance is noted in particular when the average document length is smaller. Since the length of each document is proportional to k_d, the number of topics in document d, it is easy to see that our $O(k_d)$ alias sampler is advantageous when evaluating short documents.

Figure 2 gives the comparison between PDP, AliasPDP, HDP, and AliasHDP on GPOL and Enron datasets. A substantial amount of gain in performance is noted for both PDP compared to AliasPDP, and HDP compared to AliasHDP. When AliasHDP and AliasPDP nearly converges, HDP is still at the first 1 or 2 iterations, and PDP is still at first 3 to 5 iterations.

In the following sections we evaluate the performance in two separate parts: perplexity performance over iterations, and running time over iterations.

4.3.2 *Perplexity performance over iteration*

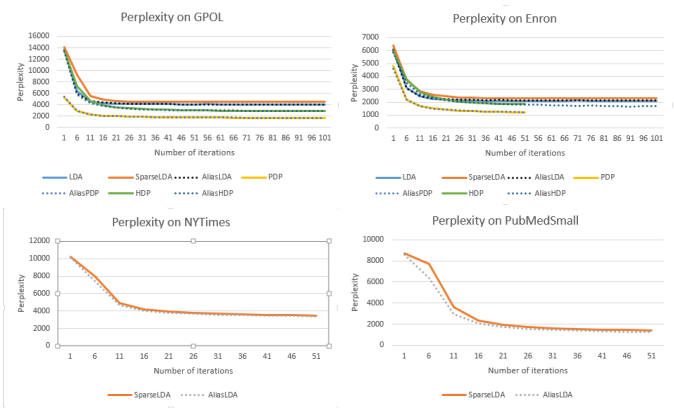

Figure 3: Perplexity along number of iterations for various methods.

The gain in performance is at no cost of degraded topic quality, as shown in Figure 3. The convergence speed and converged perplexity of AliasLDA, AliasPDP, and AliasHDP almost perfectly match the non-alias counterparts. This further shows our choice of relatively small Metropolis-Hasting steps (2 per sample) is adequate.

4.3.3 *Running time over iteration*

The better performance in running time of our alias implementations can be seen in all phases of sampling when compared to non-alias standard implementations (LDA, PDP, HDP). When compared to SparseLDA (Figure 4), the performance gain is salient during all phases on larger datasets (except for the early stage in Enron dataset), and the performance is very close on small datasets (0.35s per iteration on AliasLDA v.s 0.25 iteration on SparseLDA). As the size of the data grows AliasLDA outperforms SparseLDA without degrading topic quality, reducing the amount of time for each Gibbs iteration on NYTimes corpus by around 12% to 38% overall, on Enron corpus by around 30% after 30 iterations, and on PubMedSmall corpus by 27%-60% throughout the first 50 iterations. Compared to SparseLDA, the time required for each Gibbs iteration with AliasLDA grows at a much slower rate, and the benefits of reduced sampling complexity is particularly clear when the average length of each document is small.

The gap in performance is especially large when comparing AliasPDP v.s PDP, and AliasHDP vs HDP, as seen in 5. The running time for each Gibbs iteration is reduced by 60% to 80% for PDP, and 80% to 95% for HDP, an order of magnitude on improvement.

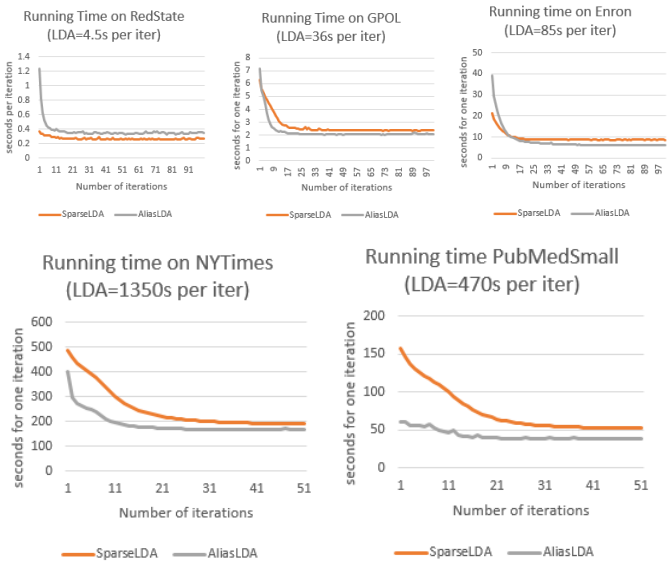

Figure 4: Running time comparison of LDA, SparseLDA, and AliasLDA

4.3.4 *The effect of varying the number of topics*

When the number of topics k increases, the running time for an iteration of AliasLDA increases at a much lower rate than SparseLDA, as seen from Figure 6 on dataset GPOL. Even though the gap between SparseLDA and AliasLDA may seem not substantial at $k = 1024$, it becomes clear at $k = 2048$ that the speed gain stays stably at around 45%. At $k = 4096$, the speed gain is over 100%. This confirms with the observation above that shorter documents benefits more from AliasLDA in the sense that the average documents length L/D relative the number of topics k becomes relatively much "shorter" as k increases, of which resulting a more sparse n_{dt} and lower k_d for a document d in average.

The result also suggests that given appropriate k, when the number of topics grows to ck for some c, the running speed of AliasLDA may grow only in $log(c)$ whereas the running speed for SparseLDA grows linearly in c. The correctness of this hypothesis and whether it can be formally proven true or false is left as for future work.

Figure 5: Running time comparison between PDP and AliasPDP, and between HDP and AliasHDP

4.3.5 *The effect on variation of corpus collection size*

Figure 7 demonstrates how the gap in running time speed scales with growing number of documents in the same domain. We measure the average runtime for the first 50 Gibbs iterations on 10%, 20%, 40%, 75%, and 100% of PubMedSmall dataset. The speedup ratio for each subset is at 31%, 34%, 37%, 41%, 43% respectively, increases along with the increasing amount of data, which conforms our intuition at section 1 that adding new documents increases the density of n_{tw} (number of topics having word w) more than n_{dt} (average number of topics per documents) that our algorithm depends on.

5. CONCLUSION

In this paper, we described an approach that effectively reduced sampling complexity of topic models from $O(k)$ to $O(k_d)$ or $O(k_w)$ in general, and from $O(k_w + k_d)$ (SparseLDA) to $O(k_d)$ (AliasLDA) for LDA topic model. Emperically, we have shown that our approach scales better than existing state-of-the-art method when the number

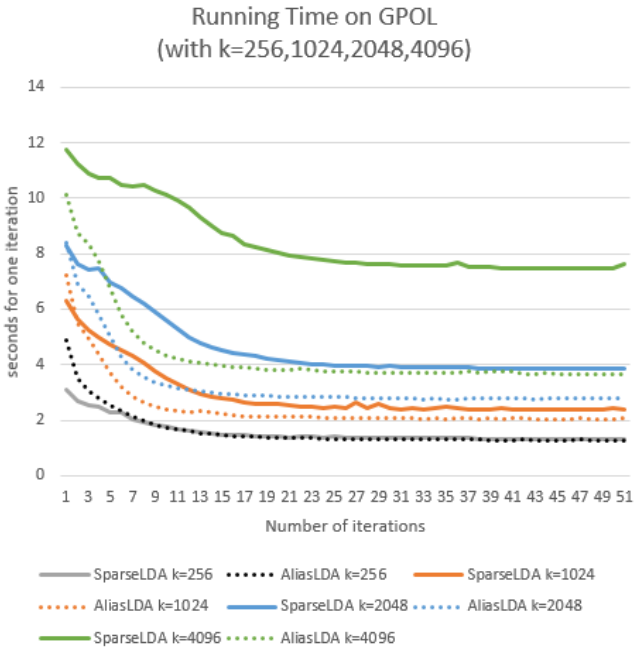

Figure 6: Varying number of topics: running time comparison with k=256,1024,2048,4096 on GPOL between SparseLDA and AliasLDA

of topics and the number of documents become large. This enables many large scale applications, and many existing applications which require distributed approach or expensive hardware such as computing clusters may now run on a single machine. In many industrial applications where the number of tokens often easily reach billions in number, these properties are crucial and often desirable in designing a scalable and responsive service. We also demonstrated an order of magnitude improvement when our approach is applied to complex models such as PDP and HDP. With an order of magnitude gain in speed, PDP and HDP may become much more appealing to many applications for their superior convergence performance, and much richer representation of topic mixtures and language models.

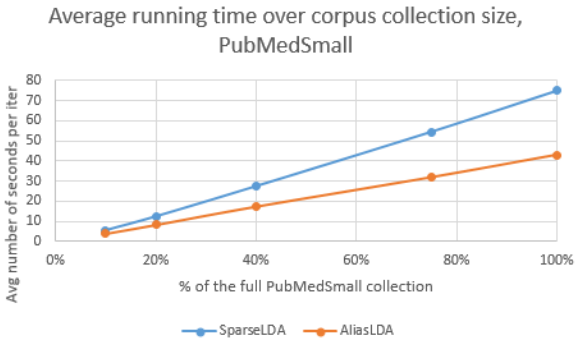

Figure 7: Average running time per iteration comparison on 10%,20%,40%,75% of PubMedSmall dataset between SparseLDA and AliasLDA

Under the settings with $k = 1024$, the number of tokens processed per second in our implementation is beyond 1 million for all datasets except one (NYTimes), of which contains substantially more lengthy documents. This is a result much beyond many existing systems and known implementations when measured in number of tokens processed per computing second per core , such as YahooLDA [18], GraphLab, and related work such as [10], given that we only utilise a single thread on a single laptop CPU core.

Acknowledgments: This work was supported in part by a resource grant from amazon.com and a Faculty Research Grant from Google.

6. REFERENCES

[1] K. Bache and M. Lichman. UCI machine learning repository, 2013.

[2] D. Blei, T. Griffiths, and M. Jordan. The nested chinese restaurant process and Bayesian nonparametric inference of topic hierarchies. *Journal of the ACM*, 57(2):1–30, 2010.

[3] D. Blei, A. Ng, and M. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, Jan. 2003.

[4] W. Buntine and M. Hutter. A bayesian review of the poisson-dirichlet process, July 2010.

[5] N. D. L. X. L. D. Changyou Chen, Wray Buntine. Differential topic models. In *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2014.

[6] C. Chen, L. Du, and W. L. Buntine. Sampling table configurations for the hierarchical poisson-dirichlet process. In D. Gunopulos, T. Hofmann, D. Malerba, and M. Vazirgiannis, editors, *ECML/PKDD (1)*, volume 6911 of *Lecture Notes in Computer Science*, pages 296–311. Springer, 2011.

[7] J. Geweke and H. Tanizaki. Bayesian estimation of state-space model using the metropolis-hastings algorithm within gibbs sampling. *Computational Statistics and Data Analysis*, 37(2):151–170, 2001.

[8] T. Griffiths and M. Steyvers. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101:5228–5235, 2004.

[9] G. Heinrich. Parameter estimation for text analysis. Technical report, 2004.

[10] Q. Ho, J. Cipar, H. Cui, S. Lee, J. Kim, P. Gibbons, G. Gibson, G. Ganger, and E. Xing. More effective distributed ml via a stale synchronous parallel parameter server. In *NIPS*, 2013.

[11] M. Hoffman, D. M. Blei, C. Wang, and J. Paisley. Stochastic variational inference. In *International Conference on Machine Learning*, 2012.

[12] W. Li, D. Blei, and A. McCallum. Nonparametric bayes pachinko allocation. In *Uncertainty in Artificial Intelligence*, 2007.

[13] G. Marsaglia, W. W. Tsang, and J. Wang. Fast generation of discrete random variables. *Journal of Statistical Software*, 11(3):1–8, 2004.

[14] R. M. Neal. Assessing relevance determination methods using delve. In C. M. Bishop, editor, *Neural Networks and Machine Learning*, pages 97–129. Springer, 1998.

[15] J. Petterson, A. Smola, T. Caetano, W. Buntine, and S. Narayanamurthy. Word features for latent dirichlet allocation. pages 1921–1929, 2010.

[16] J. Pitman and M. Yor. The two-parameter poisson-dirichlet distribution derived from a stable subordinator. *Annals of Probability*, 25(2):855–900, 1997.

[17] I. Sato and H. Nakagawa. Topic models with power-law using pitman-yor process. In B. Rao, B. Krishnapuram, A. Tomkins, and Q. Y. 0001, editors, *KDD*, pages 673–682. ACM, 2010.

[18] A. J. Smola and S. Narayanamurthy. An architecture for parallel topic models. In *Very Large Databases (VLDB)*, 2010.

[19] Y. Teh, M. Jordan, M. Beal, and D. Blei. Hierarchical dirichlet processes. *Journal of the American Statistical Association*, 101(576):1566–1581, 2006.

[20] A. J. Walker. An efficient method for generating discrete random variables with general distributions. *ACM Transactions on Mathematical Software (TOMS)*, 3(3):253–256, 1977.

[21] C. Wang, J. Paisley, and D. M. Blei. Online variational inference for the hierarchical dirichlet process. In *Proc. Intl. Conference on Artificial Intelligence and Statistics*, 2011.

[22] L. Yao, D. Mimno, and A. McCallum. Efficient methods for topic model inference on streaming document collections. In *KDD'09*, 2009.