Recap: The LDA Topic Model

Latent Dirichlet Allocation

David M. Blei
Computer Science Division
University of California
Berkeley, CA 94720, USA

Andrew Y. Ng
Computer Science Department
Stanford University
Stanford, CA 94305, USA

Michael I. Jordan
Computer Science Division and Department of Statistics
University of California
Berkeley, CA 94720, USA
Unsupervised NB vs LDA

- One class prior
- One Y per doc

- Different class distrib θ for each doc
- One Z per word

- α
- π
- Y
- W
- N_d
- D
- β
- Y
- K

- α
- θ_d
- Z_{di}
- W_{di}
- N_d
- D
- β
- Y_k
- K
- **LDA topics: top words w by Pr(w|Z=k)**

<table>
<thead>
<tr>
<th></th>
<th>“Arts”</th>
<th>“Budgets”</th>
<th>“Children”</th>
<th>“Education”</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z=13</td>
<td>NEW, FILM, SHOW, MUSIC, MOVIE, PLAY, MUSICAL, BEST, ACTOR, FIRST, YORK, OPERA, THEATER, ACTRESS, LOVE</td>
<td>MILLION, TAX, PROGRAM, BUDGET, BILLION, FEDERAL, YEAR, SPENDING, NEW, STATE, PLAN, MONEY, PROGRAMS, GOVERNMENT, CONGRESS</td>
<td>CHILDREN, WOMEN, PEOPLE, CHILD, YEARS, FAMILIES, WORK, PARENTS, SAYS, FAMILY, WELFARE, MEN, PERCENT, CARE, LIFE</td>
<td>SCHOOL, STUDENTS, SCHOOLS, EDUCATION, TEACHERS, HIGH, PUBLIC, TEACHER, BENNETT, MANIGAT, NAMPHY, STATE, PRESIDENT, ELEMENTARY, HAITI</td>
</tr>
</tbody>
</table>
The William Randolph Hearst Foundation will give $1.25 million to Lincoln Center, Metropolitan Opera Co., New York Philharmonic and Juilliard School. “Our board felt that we had a real opportunity to make a mark on the future of the performing arts with these grants an act every bit as important as our traditional areas of support in health, medical research, education and the social services,” Hearst Foundation President Randolph A. Hearst said Monday in announcing the grants. Lincoln Center’s share will be $200,000 for its new building, which will house young artists and provide new public facilities. The Metropolitan Opera Co. and New York Philharmonic will receive $400,000 each. The Juilliard School, where music and the performing arts are taught, will get $250,000. The Hearst Foundation, a leading supporter of the Lincoln Center Consolidated Corporate Fund, will make its usual annual $100,000 donation, too.
LDA and (Collapsed) Gibbs Sampling

• Gibbs sampling – works for any directed model!
  - Applicable when joint distribution is hard to evaluate but conditional distribution is known
  - Sequence of samples comprises a Markov Chain
  - Stationary distribution of the chain is the joint distribution

1. Initialise $x_{0,1:n}$.
2. For $i = 0$ to $N - 1$
   - Sample $x_1^{(i+1)} \sim p(x_1|x_2^{(i)}, x_3^{(i)}, \ldots, x_n^{(i)})$.
   - Sample $x_2^{(i+1)} \sim p(x_2|x_1^{(i+1)}, x_3^{(i)}, \ldots, x_n^{(i)})$.
     
     \[ \vdots \]
   - Sample $x_j^{(i+1)} \sim p(x_j|x_1^{(i+1)}, \ldots, x_{j-1}^{(i+1)}, x_{j+1}^{(i)}, \ldots, x_n^{(i)})$.
     
     \[ \vdots \]
   - Sample $x_n^{(i+1)} \sim p(x_n|x_1^{(i+1)}, x_2^{(i+1)}, \ldots, x_{n-1}^{(i+1)})$.  

Key capability: estimate distribution of one latent variables given the other latent variables and observed variables.
Recap: Collapsed Sampling for LDA

\[ P(z = t|w) \propto (\alpha_t + n_{t|d}) \frac{\beta + n_{w|t}}{\beta V + n_{.|t}}. \]

Pr(\(Z|E^+\)) \hspace{1cm} \text{Pr}(E^-|Z)

"fraction" of time \(Z=t\) in doc \(d\)

fraction of time \(W=w\) in topic \(t\)

ignores a detail – counts should not include the \(Z_{di}\) being sampled

Only sample the Z’s
Speeding up LDA

- Parallelize it
- Speed up sampling
PARALLEL LDA
Distributed Algorithms for Topic Models

David Newman  
Arthur Asuncion  
Padhraic Smyth  
Max Welling

Department of Computer Science  
University of California, Irvine  
Irvine, CA 92697, USA

NEWMAN@UCI.EDU  
ASUNCION@ICS.EDU  
SMYTH@ICS.EDU  
WELLING@ICS.EDU

JMLR 2009
Observation

- How much does the choice of $z$ depend on the other $z$’s in the same document?
  - quite a lot
- How much does the choice of $z$ depend on the other $z$’s in elsewhere in the corpus?
  - maybe not so much
  - depends on $Pr(w|t)$ but that changes slowly
- Can we parallelize Gibbs and still get good results?
  - formally, no: every choice of $z$ depends on all the other $z$’s
  - Gibbs needs to be sequential, just like SGD
What if you try and parallelize?

Split document/term matrix randomly and distribute to \( p \) processors. Then run “Approximate Distributed LDA”

**Algorithm 1 AD-LDA**

```
repeat
    for each processor \( p \) in parallel do
        Copy global counts: \( N_{wpk} \leftarrow N_{wk} \)
        Sample \( z_p \) locally: LDA-Gibbs-Iteration(\( x_p, z_p, N_{kpj}, N_{wpk}, \alpha, \beta \))
    end for
    Synchronize
    Update global counts: \( N_{wk} \leftarrow N_{wk} + \sum_p (N_{wpk} - N_{wk}) \)
until termination criterion satisfied
```

This is iterative parameter mixing

Let the local counters diverge
What if you try and parallelize?

D=\#docs W=\#word(types) K=\#topics N=words in corpus

<table>
<thead>
<tr>
<th></th>
<th>LDA</th>
<th>AD-LDA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Space</td>
<td>(N + K(D + W))</td>
<td>(\frac{1}{P}(N + KD) + KW)</td>
</tr>
<tr>
<td>Time</td>
<td>(NK)</td>
<td>(\frac{1}{P}NK + KW + C)</td>
</tr>
</tbody>
</table>

Table 3: Space and time complexity of LDA and AD-LDA.

Table 2: Characteristics of data sets used in experiments.
Figure 6: AD-LDA test perplexity versus number of processors up to the limiting case of number of processors equal to number of documents in collection. Left plot shows perplexity for KOS and right plot shows perplexity for NIPS.
perplexity – NIPS dataset
Figure 12: Parallel speedup results for 64 to 1024 processors on multi-million document data sets WIKIPEDIA and PUBMED.
Figure 16: Test perplexity versus iteration where synchronizations between processors only occur every 100 iterations, KOS, $K = 16$.  

match topics by similarity not topic id
Speeding up LDA

• Parallelize it
• Speed up sampling
More detail

RECAP

linear in corpus size and #topics

def initGibbs(self):
    print '.initializing latent vars'
    self.totalTopicCount = self.topicCounter()
    self.docTopicCount = [self.topicCounter() for d in xrange(len(self.x))]
    self.wordTopicCount = [self.topicCounter() for w in xrange(len(self.vocab))]
    self.z = [[-1 for j in xrange(len(self.x[d]))] for d in xrange(len(self.x))]
    for d in xrange(len(self.x)):
        if (d+1)%self.dstep==0: print '.doc',d+1,'of',len(self.x)
        for j in xrange(len(self.x[d])):
            w = self.x[d][j]
            k = random.randint(0, self.numTopics-1)
            self.z[d][j] = k
            self.docTopicCount[d].add(k, 1)
            self.wordTopicCount[w].add(k, 1)
            self.totalTopicCount.add(k, 1)

    #reasonable parameters
    self.alpha = 1.0/self.numTopics
    self.beta = 1.0/len(self.vocab)
    print "alpha:", self.alpha, "beta:", self.beta
def runGibbs(self, maxT):
    for t in xrange(maxT):
        print '.iteration', t+1, 'of', maxT
        for d in xrange(len(self.x)):
            if (d+1) % self.dstep == 0:
                print '...doc', d+1, 'of', len(self.x)
            for j in xrange(len(self.x[d])):
                k = self.resample(d, j)
                self.flip(d, j, self.z[d][j], k)

def resample(self, d, j):
    """sample a new value of z[d][j]""
    p = []
    norm = 0.0
    #compute pk = Pr(z_dj=k | everything else)
    for k in xrange(self.numTopics):
        w = self.x[d][j]
        z_dj_equals_k = 1 if self.z[d][j] == k else 0
        ak = (self.docTopicCount[d][k] - z_dj_equals_k + self.alpha)
        bk = ((self.wordTopicCount[w][k] - z_dj_equals_k + self.beta) / (self.totalTopicCount[k] - z_dj_equals_k + self.totalWords * self.beta))
        pk = ak * bk
        p.append(pk)
        norm += pk
    #pick randomly from the normalized pk
    sum_p_up_to_k = 0.0
    r = random.random()
    for k in xrange(self.numTopics):
        sum_p_up_to_k += p[k] / norm
        if r < sum_p_up_to_k:
            return k


RECAP

- Each iteration: linear in corpus size
- Resample: linear in #topics
- Most of the time is resampling
\[ P(z = t|w) \propto (\alpha_t + n_{t|d}) \frac{\beta + n_{w|t}}{\beta V + n_{.|t}}. \]

```python
def resample(self, d, j):
    """sample a new value of z[d][j]""
    p = []
norm = 0.0
    #compute pk = Pr(z_dj=k | everything else)
    for k in xrange(self.numTopics):
        w = self.x[d][j]
z_dj.equals_k = 1 if self.z[d][j]==k else 0
        #unnormalized chance of picking topic k in doc d
        ak = (self.docTopicCount[d][k] - z_dj.equals_k + self.alpha)
        #unnormalized chance of picking topic k for word w
        bk = ((self.wordTopicCount[w][k] - z_dj.equals_k + self.beta)
              / (self.totalTopicCount[k] - z_dj.equals_k + self.totalWords * self.beta))
        pk = ak*bk
        p.append(pk)
norm += pk
    #pick randomly from the normalized pk
    sum_p_up_to_k = 0.0
    r = random.random()
    for k in xrange(self.numTopics):
        sum_p_up_to_k += p[k]/norm
        if r < sum_p_up_to_k:
            return k
```

1. You spend a lot of time sampling
2. There’s a loop over all topics here in the sampler
Efficient Methods for Topic Model Inference on Streaming Document Collections

Limin Yao, David Mimno, and Andrew McCallum
Department of Computer Science
University of Massachusetts, Amherst
{lmyao, mimno, mccallum}@cs.umass.edu

KDD 09
\[ P(z = t | w) \propto (\alpha_t + n_{t|d}) \frac{\beta + n_{w|t}}{\beta V + n_{.|t}}. \]

\[ P(z = t | w) \propto \frac{\alpha_t \beta}{\beta V + n_{.|t}} + \frac{n_{t|d} \beta}{\beta V + n_{.|t}} + \frac{(\alpha_t + n_{t|d}) n_{w|t}}{\beta V + n_{.|t}}. \]
\[ P(z = t | w) \propto (\alpha_t + n_{t|d}) \frac{\beta + n_{w|t}}{\beta V + n_{.|t}}. \]

\[ P(z = t | w) \propto \frac{\alpha_t \beta}{\beta V + n_{.|t}} + \frac{n_{t|d} \beta}{\beta V + n_{.|t}} + \frac{(\alpha_t + n_{t|d}) n_{w|t}}{\beta V + n_{.|t}}. \]

\[
\begin{aligned}
  s &= \sum_t \frac{\alpha_t \beta}{\beta V + n_{.|t}} \\
  r &= \sum_t \frac{n_{t|d} \beta}{\beta V + n_{.|t}} \\
  q &= \sum_t \frac{(\alpha_t + n_{t|d}) n_{w|t}}{\beta V + n_{.|t}}.
\end{aligned}
\]

\[ z = s + r + q \]
• Draw random $U$ from uniform $[0, s+r+q]$
• If $U<s$:
  • lookup $U$ on line segment with tic-marks at $\alpha_1\beta/(\beta V + n_{1t}), \alpha_2\beta/(\beta V + n_{2t}), ...$

$$s = \sum_t \frac{\alpha_t\beta}{\beta V + n_{1t}}$$

$$r$$

$$q$$

ormalizer = $s+r+q$
• If $U<s$:
  • lookup $U$ on line segment with tic-marks at $\alpha_1 \beta / (\beta V + n_{|1})$, $\alpha_2 \beta / (\beta V + n_{|2})$, ...
• If $s<U<s+r$:
  • lookup $U$ on line segment for $r$

\[
\begin{align*}
  z &= s + r + q \\
  s &= \sum_t \frac{\alpha_t \beta}{\beta V + n_{|t}} \\
  r &= \sum_t \frac{n_{t|d} \beta}{\beta V + n_{|t}} \\
  q &= \sum_t \frac{(\alpha_t + n_{t|d}) n_{w|t}}{\beta V + n_{|t}}.
\end{align*}
\]
• If $U < s$:
  • lookup $U$ on line segment with tic-marks at $\alpha_1 \beta / (\beta V + n_{|1})$, $\alpha_2 \beta / (\beta V + n_{|2})$, ...
• If $s < U < s + r$:
  • lookup $U$ on line segment for $r$
• If $s + r < U$:
  • lookup $U$ on line segment for $q$

\[
\begin{align*}
  z &= s + r + q \\
  r &= \sum_t \frac{n_t|d/\beta}{\beta V + n_{|t}} \\
  q &= \sum_t \frac{(\alpha_t + n_t|d)n_{w|t}}{\beta V + n_{|t}}.
\end{align*}
\]

Only need to check $t$ such that $n_{w|t} > 0$. 

\[ z = s + r + q \]

\[ s = \sum_t \frac{\alpha_t \beta}{\beta V + n_t|t} \]

\[ r = \sum_t \frac{n_t|d \beta}{\beta V + n_t|t} \]

\[ q = \sum_t \frac{(\alpha_t + n_t|d)n_w|t}{\beta V + n_t|t} \]

Only need to check occasionally (< 10% of the time)

Only need to check \( t \) such that \( n_t|d > 0 \)

Only need to check \( t \) such that \( n_w|t > 0 \)
\[ z = s + r + q \]

\[ s = \sum_t \frac{\alpha_t \beta}{\beta V + n_{.t}} \]

\[ r = \sum_t \frac{n_{t.d} \beta}{\beta V + n_{.t}} \]

\[ q = \sum_t \frac{(\alpha_t + n_{t.d}) n_{w.t}}{\beta V + n_{.t}} \]

Need to store \( n_{w.t} \) for each word, topic pair …???

Only need to store \( n_{t.d} \) for current \( d \) and update incrementally

Only need to store \( \alpha, \beta, V \) (and maintain) total words per topic and topic pair…???
\[ q = \sum_t \left[ \frac{\alpha_t + n_{t|d}}{\beta V + n._t} \right] \times n_{w|t} \].

1. Maintain, for \( d \) and each \( t \),

2. Quickly find \( t \)'s such that \( n_{w|t} \) is large for \( w \)

\[
\begin{align*}
1. & \quad \text{Maintain, for } d \text{ and each } t, \\
2. & \quad \text{Quickly find } t \text{'s such that } n_{w|t} \text{ is large for } w
\end{align*}
\]

Most (>90%) of the time and space is here...

Need to store \( n_{w|t} \) for each word, topic pair ...???

\[
\begin{align*}
q &= \sum_t \frac{\alpha_t + n_{t|d}}{\beta V + n._t} n_{w|t} \\
r &= \sum_t \frac{n_{t|d} \beta}{\beta V + n._t} \\
s &= \sum_t \frac{\alpha_t \beta}{\beta V + n._t} \\
\end{align*}
\]
Topic distributions are skewed!

Topic proportions learned by LDA on a typical NIPS paper
\[ q = \sum_t \left[ \frac{\alpha_t + n_{t|d}}{\beta V + n_{.|t}} \times n_{w|t} \right]. \]

1. **Precompute, for each** \( t \), \[ \frac{\alpha_t + n_{t|d}}{\beta V + n_{.|t}} \]

2. **Quickly find** \( t \)'s such that \( n_{w|t} \) is large for \( w \)

Here’s how Mimno did it:
- associate each \( w \) with an int array
  - no larger than frequency of \( w \)
  - no larger than #topics
- encode \((t,n)\) as a bit vector
  - \( n \) in the high-order bits
  - \( t \) in the low-order bits
- keep ints sorted in descending order

Most (>90%) of the time and space is here…

Need to **store** \( n_{w|t} \) for each word, topic pair …???
Figure 2: A comparison of time and space efficiency between standard Gibbs sampling (dashed red lines) and the SparseLDA algorithm and data structure presented in this paper (solid black lines). Error bars show the standard deviation over five runs.
Other Fast Samplers for LDA
Alias tables

Basic problem: how can we sample from a biased coin quickly?

If the distribution changes slowly maybe we can do some preprocessing and then sample multiple times. Proof of concept: generate $r \sim \text{uniform}$ and use a binary tree

http://www.keithschwarz.com/darts-dice-coins/
Alias tables

Basic problem: how can we sample from a biased die quickly?

Naively this is $O(K)$
Another idea…

Simulate the dart with two drawn values:

\[ rx \rightarrow \text{int}(u_1 \times K) \]
\[ ry \rightarrow u_1 \times p_{\text{max}} \]

keep throwing till you hit a stripe
An even more clever idea: minimize the brown space (where the dart “misses”) by sizing the rectangle’s height to the average probability, not the maximum probability, and cutting and pasting a bit.

You can always do this using only two colors in each column of the final alias table and the dart never misses!

mathematically speaking…
LDA with Alias Sampling

Reducing the Sampling Complexity of Topic Models

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

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

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

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

• Sample Z’s with alias sampler
• Don’t update the sampler with each flip:
  – Correct for “staleness” with Metropolis-Hastings algorithm
GPOL
(36s per LDA iteration)
Yet More Fast Samplers for LDA
A Scalable Asynchronous Distributed Algorithm for Topic Modeling

Hsiang-Fu Yu
Univ. of Texas, Austin
rofuyu@cs.utexas.edu

Cho-Jui Hsieh
Univ. of Texas, Austin
cjhsieh@cs.utexas.edu

Hyokun Yun
Amazon
yunhyoku@amazon.com

S.V.N Vishwanathan
Univ. of California, Santa Cruz
vishy@ucsc.edu

Inderjit S. Dhillon
Univ. of Texas, Austin
inderjit@cs.utexas.edu

WWW 2015
Fenwick Tree (1994)

Basic problem: how can we sample from a biased die quickly…. 

…and update quickly? maybe we can use a binary tree…. 

http://www.keithschwarz.com/darts-dice-coins/
# Data structures and algorithms

## LSearch: linear search

### Table

<table>
<thead>
<tr>
<th>Data Structure</th>
<th>Initialization</th>
<th>Generation</th>
<th>Parameter Update</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSearch</td>
<td>( c_T = p^\top 1: O(1) )</td>
<td>( O(T) )</td>
<td>( O(1) )</td>
</tr>
<tr>
<td>BSearch</td>
<td>( c = \text{cumsum}(p): O(T) )</td>
<td>( O(T) )</td>
<td>( O(1) )</td>
</tr>
<tr>
<td>Alias Method</td>
<td>( \text{prob}, \text{alias}: O(T) )</td>
<td>( O(T) )</td>
<td>( O(T) )</td>
</tr>
<tr>
<td>F+tree Sampling</td>
<td>( \text{F.initialize}(p): O(T) )</td>
<td>( O(T) )</td>
<td>( O(1) )</td>
</tr>
</tbody>
</table>
Data structures and algorithms

<table>
<thead>
<tr>
<th>Data Structure</th>
<th>Initialization</th>
<th>Generation</th>
<th>Parameter Update</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Space</td>
<td>Time</td>
<td>Space</td>
</tr>
<tr>
<td>LSearch</td>
<td>$c_T = p^1$: $O(1)$</td>
<td>$O(T)$</td>
<td>$O(1)$</td>
</tr>
<tr>
<td>BSearch</td>
<td>$c = \text{cumsum}(p)$: $O(T)$</td>
<td>$O(T)$</td>
<td>$O(1)$</td>
</tr>
<tr>
<td>Alias Method</td>
<td>$\text{prob, alias}$: $O(T)$</td>
<td>$O(T)$</td>
<td>$O(T)$</td>
</tr>
<tr>
<td>F+tree Sampling</td>
<td>$\text{F.initialize}(p)$: $O(T)$</td>
<td>$O(T)$</td>
<td>$O(1)$</td>
</tr>
</tbody>
</table>

BSearch: binary search

store cumulative probability
Data structures and algorithms

<table>
<thead>
<tr>
<th>Data Structure</th>
<th>Initialization</th>
<th>Generation</th>
<th>Parameter Update</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSearch</td>
<td>$c_T = p^\top 1: O(1)$</td>
<td>$O(T)$</td>
<td>$O(T)$</td>
</tr>
<tr>
<td>BSearch</td>
<td>$c = \text{cumsum}(p): O(T)$</td>
<td>$O(T)$</td>
<td>$O(\log T)$</td>
</tr>
<tr>
<td>Alias Method</td>
<td>$\text{prob}, \text{alias}: O(T)$</td>
<td>$O(T)$</td>
<td>$O(1)$</td>
</tr>
<tr>
<td>F+tree Sampling</td>
<td>$\text{F.initialize}(p): O(T)$</td>
<td>$O(T)$</td>
<td>$O(\log T)$</td>
</tr>
</tbody>
</table>

Alias sampling.....

- [Image of a colored diagram with fractions representing probabilities]
Data structures and algorithms

<table>
<thead>
<tr>
<th>Data Structure</th>
<th>Initialization</th>
<th>Generation</th>
<th>Parameter Update</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time</td>
<td>Space</td>
<td>Time</td>
</tr>
<tr>
<td>LSearch</td>
<td>$c_T = p^1$</td>
<td>$O(1)$</td>
<td>$O(T)$</td>
</tr>
<tr>
<td>BSearch</td>
<td>$c = \text{cumsum}(p)$</td>
<td>$O(T)$</td>
<td>$O(1)$</td>
</tr>
<tr>
<td>Alias Method</td>
<td>$\text{prob, alias: } O(T)$</td>
<td>$O(T)$</td>
<td>$O(1)$</td>
</tr>
<tr>
<td>F+tree Sampling</td>
<td>$\text{F.initialize}(p): O(T)$</td>
<td>$O(T)$</td>
<td>$O(1)$</td>
</tr>
</tbody>
</table>

F+ tree
Data structures and algorithms

\[ p_t = \frac{(n_{td} + \alpha)(n_{tw} + \beta)}{n_t + \beta}, \quad \forall t = 1, \ldots, T. \]

\[ = \beta \left( \frac{n_{td} + \alpha}{n_t + \beta} \right) + n_{tw} \left( \frac{n_{td} + \alpha}{n_t + \beta} \right). \]

\[ \beta q: \text{dense, changes slowly, re-used for each word in a document} \]

\[ r: \text{sparse, a different one is needed for each unique term in the document} \]

Sampler is:

\[ \text{discrete}(p, u) = \begin{cases} \text{discrete}(r, u) & \text{if } u \leq r^T 1, \\ \text{discrete}(q, \frac{u - r^T 1}{\beta}) & \text{otherwise}, \end{cases} \]
Enron -- normal LDA (44s per iter)

Nytimes -- normal LDA (730s per iter)

Speedup vs std LDA sampler (1024 topics)

Table 3: Data statistics.

<table>
<thead>
<tr>
<th></th>
<th># documents (I)</th>
<th># vocabulary (J)</th>
<th># words</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enron</td>
<td>37,861</td>
<td>28,102</td>
<td>6,238,796</td>
</tr>
<tr>
<td>NyTimes</td>
<td>298,000</td>
<td>102,660</td>
<td>98,793,316</td>
</tr>
<tr>
<td>PubMed</td>
<td>8,200,000</td>
<td>141,043</td>
<td>737,869,083</td>
</tr>
<tr>
<td>Amazon</td>
<td>29,907,995</td>
<td>1,682,527</td>
<td>1,499,602,431</td>
</tr>
<tr>
<td>UMBC</td>
<td>40,599,164</td>
<td>2,881,476</td>
<td>1,483,145,192</td>
</tr>
</tbody>
</table>
Speedup vs std LDA sampler (10k-50k topics)

Table 3: Data statistics.

<table>
<thead>
<tr>
<th></th>
<th># documents (I)</th>
<th># vocabulary (J)</th>
<th># words</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enron</td>
<td>37,861</td>
<td>28,102</td>
<td>6,238,796</td>
</tr>
<tr>
<td>NyTimes</td>
<td>298,000</td>
<td>102,660</td>
<td>98,793,316</td>
</tr>
<tr>
<td>PubMed</td>
<td>8,200,000</td>
<td>141,043</td>
<td>737,869,083</td>
</tr>
<tr>
<td>Amazon</td>
<td>29,907,995</td>
<td>1,682,527</td>
<td>1,499,602,431</td>
</tr>
<tr>
<td>UMBC</td>
<td>40,599,164</td>
<td>2,881,476</td>
<td>1,483,145,192</td>
</tr>
</tbody>
</table>
Also describe some nice ways to parallelize this operation, similar to the distributed MF algorithm we discussed.
And Parallelism....
Second idea: you can sample document-by-document or word-by-word .... or....

use a MF-like approach to distributing the data.
(a) Initial assignment of $w_j$. Each worker works only on the diagonal active area in the beginning.

(b) After a worker finishes processing $j$, it sends the corresponding $w_j$ to another worker. Here, $w_2$ is sent from worker 1 to 4.

(c) Upon receipt, the $w_j$ is processed by the new worker. Here, worker 4 can now process $w_2$ since it owns it.

(d) During the execution of the algorithm, the ownership of the $w_j$ changes.
Multi-core NOMAD method

Table 3: Data statistics.

<table>
<thead>
<tr>
<th></th>
<th># documents (I)</th>
<th># vocabulary (J)</th>
<th># words</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enron</td>
<td>37,861</td>
<td>28,102</td>
<td>6,238,796</td>
</tr>
<tr>
<td>NyTimes</td>
<td>298,000</td>
<td>102,660</td>
<td>98,793,316</td>
</tr>
<tr>
<td>PubMed</td>
<td>8,200,000</td>
<td>141,043</td>
<td>737,869,083</td>
</tr>
</tbody>
</table>
Speeding up LDA

- Parallelize it
- Speed up sampling
Speeding up LDA-like models

• Parallelize it
• Speed up sampling
• Use these tricks for other models...
Network Datasets

• UBMCBlog
• AGBlog
• MSPBlog
• Cora
• Citeseer
How do you model such graphs?

• “Stochastic block model”, aka “Block-stochastic matrix”:
  – Draw $n_i$ nodes in block $i$
  – With probability $p_{ij}$, connect pairs $(u,v)$ where $u$ is in block $i$, $v$ is in block $j$
  – Special, simple case: $p_{ii}=q_i$, and $p_{ij}=s$ for all $i \neq j$

• Question: can you fit this model to a graph?
  – find each $p_{ij}$ and latent node→block mapping
Not? books
A mixed membership stochastic block model
Stochastic Block models

Airoldi, Blei, Feinberg, Xing. JMLR 2008
Another mixed membership block model

Parkkinen, SinkkonenGyenge, Kaski, MLG 2009

\[ p(z_l | \{z\}^{-l}, \{(i, j)\}^{-l}, \alpha, \beta) \propto (n_z^{-l} + \alpha) \cdot \frac{(q_{z1i}^{-l} + \beta)(q_{z2j}^{-l} + \beta)}{(q_{z1.}^{-l} + M\beta)(q_{z2.}^{-l} + M\beta + \delta_z)}, \]
Another mixed membership block model

Pick two multinomials over nodes

For each edge in the graph:

• Pick \( z=(z_i,z_j) \), a pair of block ids
• Pick node \( i \) based on \( \text{Pr}(.)|z_i) \)
• Pick node \( j \) based on \( \text{Pr}(.)|z_j) \)
Experiments – for next week

(e) 1-NN: Social networks

<table>
<thead>
<tr>
<th>Dataset</th>
<th>PSK</th>
<th>PIC_D</th>
<th>PIC_R</th>
<th>PIC_R4</th>
<th>NCut</th>
<th>NJW</th>
</tr>
</thead>
<tbody>
<tr>
<td>Karate</td>
<td>1.00</td>
<td>1.00</td>
<td>0.99</td>
<td>0.99</td>
<td>1.00</td>
<td>0.97</td>
</tr>
<tr>
<td>Dolphin</td>
<td>0.89</td>
<td>0.95</td>
<td>0.95</td>
<td>0.95</td>
<td>0.95</td>
<td>0.98</td>
</tr>
<tr>
<td>UMBC</td>
<td>0.92</td>
<td>0.93</td>
<td>0.93</td>
<td>0.93</td>
<td>0.92</td>
<td>0.94</td>
</tr>
<tr>
<td>AG</td>
<td>0.92</td>
<td>0.94</td>
<td>0.93</td>
<td>0.93</td>
<td>0.88</td>
<td>0.89</td>
</tr>
<tr>
<td>MSP</td>
<td>0.84</td>
<td>0.76</td>
<td>0.73</td>
<td>0.86</td>
<td>0.64</td>
<td>0.59</td>
</tr>
<tr>
<td>Senate</td>
<td>0.97</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>PolBook</td>
<td>0.79</td>
<td>0.68</td>
<td>0.76</td>
<td>0.80</td>
<td>0.84</td>
<td>0.78</td>
</tr>
<tr>
<td>Football</td>
<td>0.89</td>
<td>0.43</td>
<td>0.45</td>
<td>0.85</td>
<td>0.94</td>
<td>0.95</td>
</tr>
<tr>
<td>MGEmail</td>
<td>0.22</td>
<td>0.27</td>
<td>0.26</td>
<td>0.72</td>
<td>0.80</td>
<td>0.81</td>
</tr>
<tr>
<td>CiteSeer</td>
<td>0.34</td>
<td>0.55</td>
<td>0.54</td>
<td>0.71</td>
<td>0.69</td>
<td>0.66</td>
</tr>
<tr>
<td>Cora</td>
<td>0.45</td>
<td>0.56</td>
<td>0.51</td>
<td>0.80</td>
<td>0.47</td>
<td>0.75</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td>0.75</td>
<td>0.73</td>
<td>0.73</td>
<td><strong>0.87</strong></td>
<td>0.83</td>
<td>0.85</td>
</tr>
</tbody>
</table>

(f) 1-NN: Author disambiguation

<table>
<thead>
<tr>
<th>Dataset</th>
<th>PSK</th>
<th>PIC_D</th>
<th>PIC_R</th>
<th>PIC_R4</th>
<th>NCut</th>
<th>NJW</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGupta</td>
<td>0.68</td>
<td>0.74</td>
<td>0.72</td>
<td><strong>0.95</strong></td>
<td>0.79</td>
<td>0.91</td>
</tr>
<tr>
<td>AKumar</td>
<td>0.82</td>
<td>0.69</td>
<td>0.74</td>
<td><strong>0.85</strong></td>
<td>0.79</td>
<td>0.81</td>
</tr>
<tr>
<td>CChen</td>
<td>0.77</td>
<td>0.73</td>
<td>0.74</td>
<td><strong>0.89</strong></td>
<td>0.75</td>
<td>0.85</td>
</tr>
<tr>
<td>DJohnson</td>
<td>0.81</td>
<td>0.81</td>
<td>0.83</td>
<td><strong>0.95</strong></td>
<td>0.85</td>
<td>0.92</td>
</tr>
<tr>
<td>JLee</td>
<td>0.55</td>
<td>0.61</td>
<td>0.68</td>
<td><strong>0.92</strong></td>
<td>0.79</td>
<td>0.91</td>
</tr>
<tr>
<td>JMartin</td>
<td>0.77</td>
<td>0.73</td>
<td>0.73</td>
<td><strong>0.88</strong></td>
<td>0.75</td>
<td>0.85</td>
</tr>
<tr>
<td>JRobinson</td>
<td>0.86</td>
<td>0.75</td>
<td>0.80</td>
<td><strong>0.92</strong></td>
<td>0.83</td>
<td>0.85</td>
</tr>
<tr>
<td>JSmith</td>
<td>0.65</td>
<td>0.75</td>
<td>0.67</td>
<td><strong>0.93</strong></td>
<td>0.85</td>
<td>0.91</td>
</tr>
<tr>
<td>KTanaka</td>
<td>0.81</td>
<td>0.84</td>
<td>0.86</td>
<td><strong>0.95</strong></td>
<td>0.90</td>
<td>0.90</td>
</tr>
<tr>
<td>MBrown</td>
<td>0.83</td>
<td>0.78</td>
<td>0.82</td>
<td><strong>0.93</strong></td>
<td>0.86</td>
<td>0.89</td>
</tr>
<tr>
<td>MJones</td>
<td>0.79</td>
<td>0.69</td>
<td>0.71</td>
<td><strong>0.91</strong></td>
<td>0.90</td>
<td>0.89</td>
</tr>
<tr>
<td>MMiller</td>
<td>0.81</td>
<td>0.83</td>
<td>0.81</td>
<td><strong>0.99</strong></td>
<td>0.97</td>
<td>0.98</td>
</tr>
<tr>
<td>SLee</td>
<td>0.59</td>
<td>0.69</td>
<td>0.77</td>
<td><strong>0.92</strong></td>
<td>0.85</td>
<td>0.92</td>
</tr>
<tr>
<td>YChen</td>
<td>0.57</td>
<td>0.73</td>
<td>0.79</td>
<td><strong>0.95</strong></td>
<td>0.84</td>
<td>0.94</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td>0.74</td>
<td>0.74</td>
<td>0.76</td>
<td><strong>0.92</strong></td>
<td>0.84</td>
<td>0.90</td>
</tr>
</tbody>
</table>

Balasubramanyan, Lin, Cohen, NIPS w/s 2010