

Efficient Inference for Multinomial Mixed Membership Models

David Mimno
UMass, Amherst

Summary

- How **fast** can we make Gibbs sampling?
- How little **memory** can we use?

Goal: small collections should be fast,
large collections should be possible

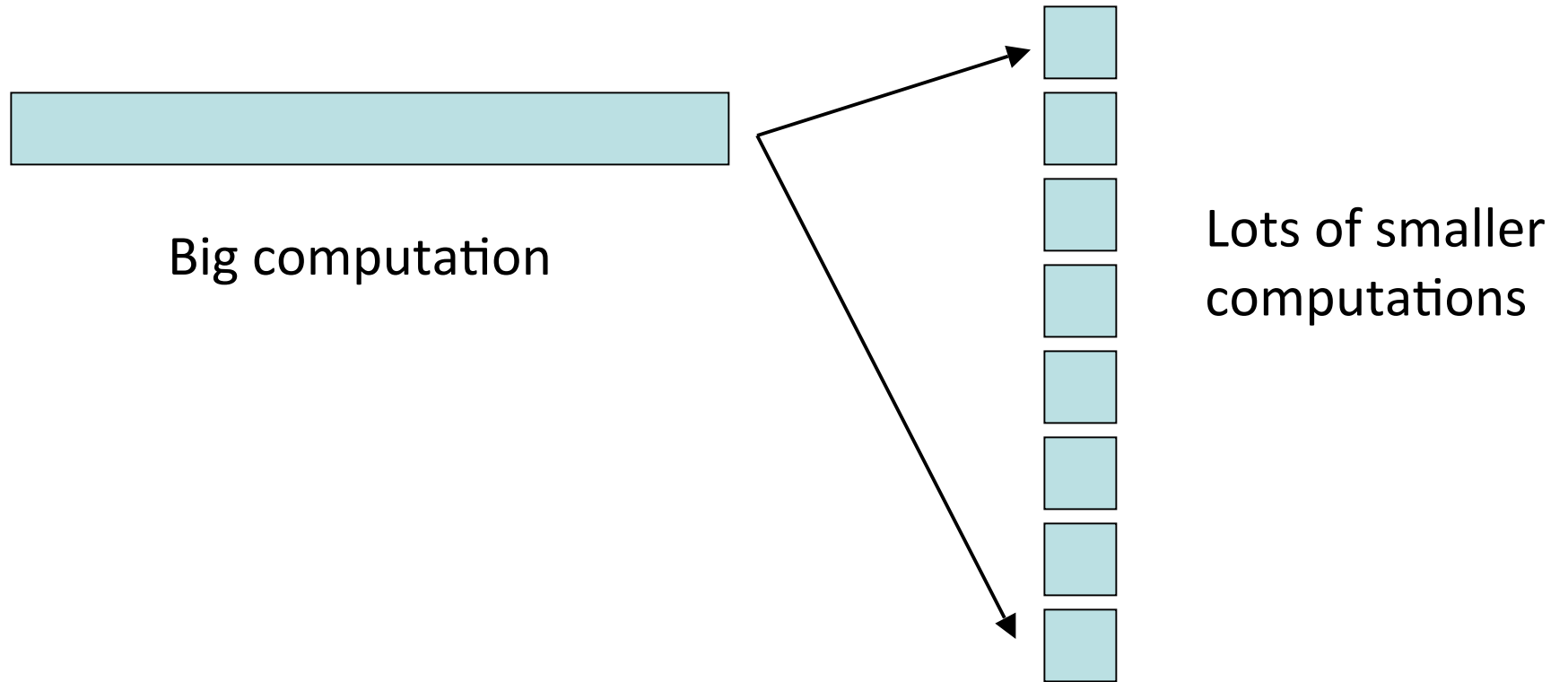
Big collections! Lots of topics!

The screenshot shows the PubMed.gov website. At the top, there is a navigation bar with "NCBI Resources" and "How To" menus, and a "My NCBI Sign In" link. Below this is the "PubMed.gov" logo and the text "U.S. National Library of Medicine National Institutes of Health". A search bar contains the text "PubMed" and has "Limits", "Advanced search", and "Help" links. A "Search" button and a "Clear" button are also present. Below the search bar, there is a "PubMed" section with a description: "PubMed comprises more than 19 million MEDLINE, life science journals, and online full-text content from PubMed Central and other sources." To the left of this section are two columns of links: "Using PubMed" (including Quick Start Guide, Full Text Articles, PubMed FAQs, PubMed Tutorials, and New and Noteworthy) and "PubMed Tools" (including Single Citation Matcher, Batch Citation Matcher, Clinical Queries, and Topic-Specific Queries). At the bottom, there is a breadcrumb trail: "You are here: NCBI > Literature > PubMed".

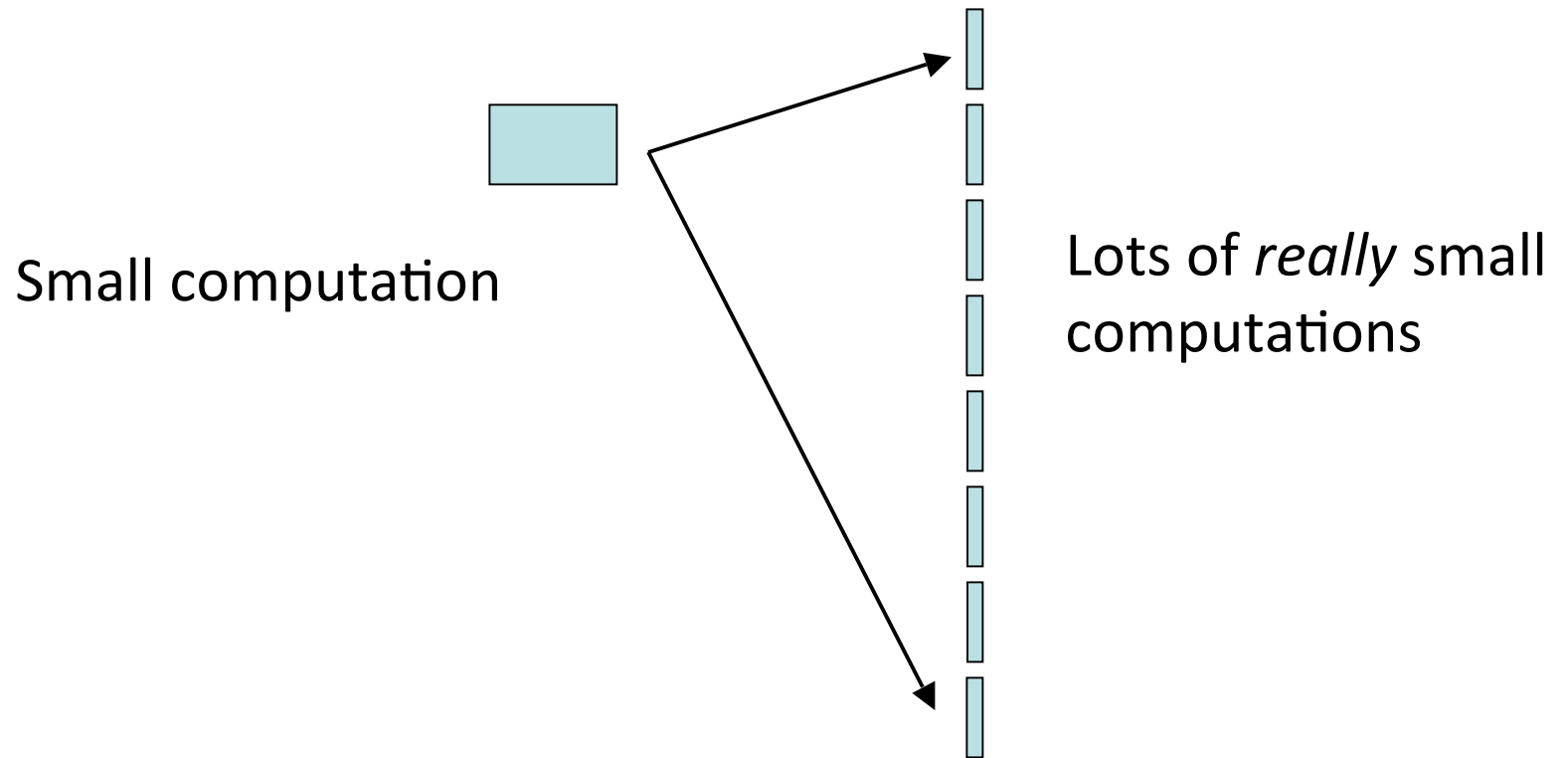
The screenshot shows the Wikipedia homepage. At the top, the word "WIKIPEDIA" is displayed in a large, serif font. Below this, there are several language options, each with a description and the number of articles: English (The Free Encyclopedia, 3 319 000+ articles), 日本語 (フリー百科事典, 682 000+ 記事), Deutsch (Die freie Enzyklopädie, 1 079 000+ Artikel), Español (La enciclopedia libre, 607 000+ artículos), Français (L'encyclopédie libre, 957 000+ articles), Italiano (L'enciclopedia libera, 696 000+ voci), Polski (Wolna encyklopedia, 706 000+ haset), and Русский (Свободная энциклопедия, 547 000+ статей). In the center, there is a globe made of puzzle pieces, each with a different symbol or character. Below the globe, there are more language options: Português (A enciclopédia livre, 585 000+ artigos) and Nederlands (De vrije encyclopedie, 606 000+ artikelen). At the bottom, there is a search bar with a magnifying glass icon, a dropdown menu set to "English", and a search button. Below the search bar, there is a list of search terms in various languages: "search • suchen • rechercher • szukaj • ricerca • 検索 • buscar • zoeken • busca • поиск • sök • 搜索 • sök • cerca • haku • пошук • hledání • keresés • ara • căutare • 찾기 • serchu • søg • بحث • cari • tìm kiếm • suk • претрага • hľadat • paieška • חיפוש • търсене". At the very bottom, there is a footer with a list of languages: "العربية • Български • Català • Český • Dansk • Deutsch • English • Español • Esperanto • Français • 한국어 •".

- 1000+ topics

Why not just parallelize?



Why not just parallelize?



Be smart, then parallelize

Gibbs Sampling for Topic Models

Doc A

dog	0
cat	2
feline	2
hound	0

Doc B

dog	0
dog	0
horse	1
equine	1

Document-topic statistics

Doc A

dog	0
cat	2
feline	2
hound	0

Doc B

dog	0
dog	0
horse	1
equine	1

topics \longrightarrow

docs \downarrow

	0	1	2
A	2	0	2
B	2	2	0

$N_{t|d}$

Topic-word statistics

Doc A

dog	0
cat	2
feline	2
hound	0

Doc B

dog	0
dog	0
horse	1
equine	1

Word types \longrightarrow

Topics \downarrow

	dog	cat	horse	hound	feline
0	3			1	
1			1		
2		1			1

$N_{w|t}$

$$\text{score}(t \mid N_{t|d}, N_{w|t}) =$$

$$\frac{N_{t|d} + \alpha}{N_d + T\alpha} \times \frac{N_{w|t} + \beta}{N_t + V\beta}$$

For each token,

Doc A

dog	
cat	2
feline	2
hound	0



for each possible topic [0, 1, 2],

Doc B

dog	0
dog	0
horse	1
equine	1

For each token,

Doc A

dog	
cat	2
feline	2
hound	0

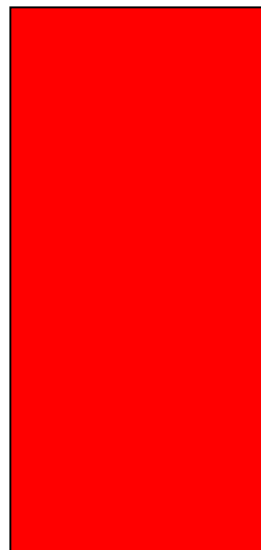


for each possible topic [0, 1, 2],
compute

$$\text{score}(0 \mid N_{0|A}, N_{\text{dog}|0})$$

Doc B

dog	0
dog	0
horse	1
equine	1



For each token,

Doc A

dog	
cat	2
feline	2
hound	0

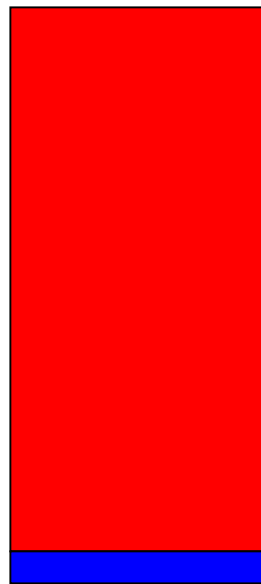


for each possible topic [0, 1, 2],
compute

$$\text{score}(0 \mid N_{0|A}, N_{\text{dog}|0})$$

Doc B

dog	0
dog	0
horse	1
equine	1



$$\text{score}(1 \mid N_{1|A}, N_{\text{dog}|1})$$

For each token,

Doc A

dog	
cat	2
feline	2
hound	0

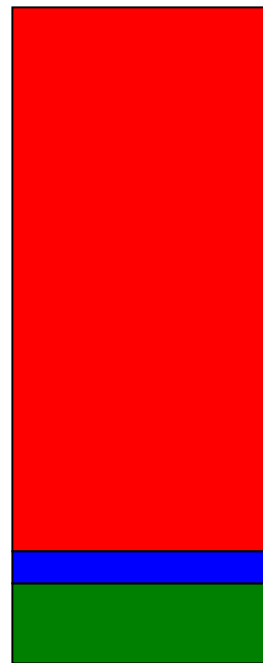


for each possible topic [0, 1, 2],
compute

$$\text{score}(0 \mid N_{0|A}, N_{\text{dog}|0})$$

Doc B

dog	0
dog	0
horse	1
equine	1



$$\text{score}(1 \mid N_{1|A}, N_{\text{dog}|1})$$

$$\text{score}(2 \mid N_{2|A}, N_{\text{dog}|2})$$

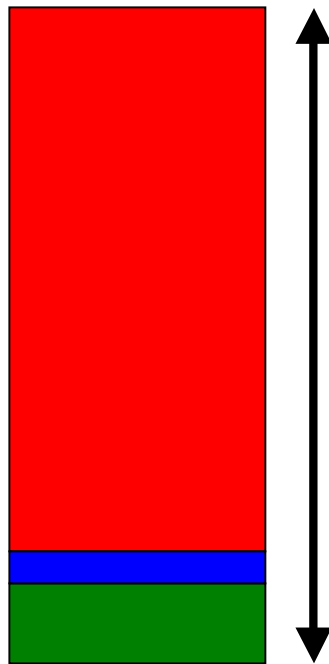
Add up the scores

Doc A

dog	
cat	2
feline	2
hound	0

Doc B

dog	0
dog	0
horse	1
equine	1



$$Z = \begin{aligned} & \text{score}(0 \mid N_{0|A}, N_{\text{dog}|0}) \\ & + \\ & \text{score}(1 \mid N_{1|A}, N_{\text{dog}|1}) \\ & + \\ & \text{score}(2 \mid N_{2|A}, N_{\text{dog}|2}) \end{aligned}$$

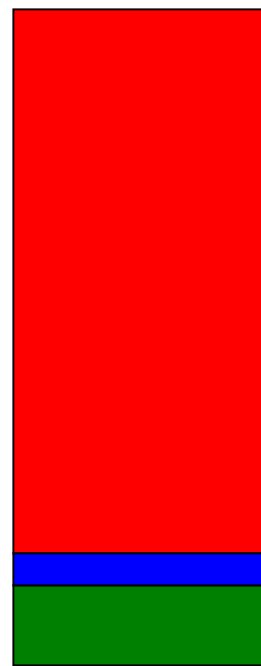
Sample a new topic

Doc A

dog	
cat	2
feline	2
hound	0

Doc B

dog	0
dog	0
horse	1
equine	1



Sample

$u = \text{rand}() * Z$

Return $t=0$

For each token,

Doc A

dog	0
cat	
feline	2
hound	0



for each possible topic [0, 1, 2],

Doc B

dog	0
dog	0
horse	1
equine	1

Summary so far

- For every token,
 - For each possible topic t calculate $\text{score}(t \mid \dots)$
- Add up scores to **normalizing constant**

$$Z = \sum_t \text{score}(t)$$

- Sample $u \sim U(0, Z)$ and return the corresponding topic t .

Performance is dominated by calculation of Z

The normalizing constant

$$Z = \sum_t \frac{N_{t|d} + \alpha}{N_d + T\alpha} \times \frac{N_{w|t} + \beta}{N_t + V\beta}$$

The normalizing constant

$$Z = \sum_t \frac{(N_{t|d} + \alpha)(N_{w|t} + \beta)}{N_t + V\beta}$$

The normalizing constant

$$Z = \sum_t \frac{N_{t|d} N_{w|t} + \beta N_{t|d} + \alpha N_{w|t} + \alpha\beta}{N_t + V\beta}$$

The normalizing constant

$$Z = \sum_t \frac{N_{w|t}(N_{t|d} + \alpha)}{N_t + V\beta} + \sum_t \frac{\beta N_{t|d}}{N_t + V\beta} + \sum_t \frac{\alpha\beta}{N_t + V\beta}$$

The normalizing constant

$$Z = \sum_t \frac{N_{w|t}(N_{t|d} + \alpha)}{N_t + V\beta} + \sum_t \frac{\beta N_{t|d}}{N_t + V\beta} + \sum_t \frac{\alpha\beta}{N_t + V\beta}$$

Token-specific

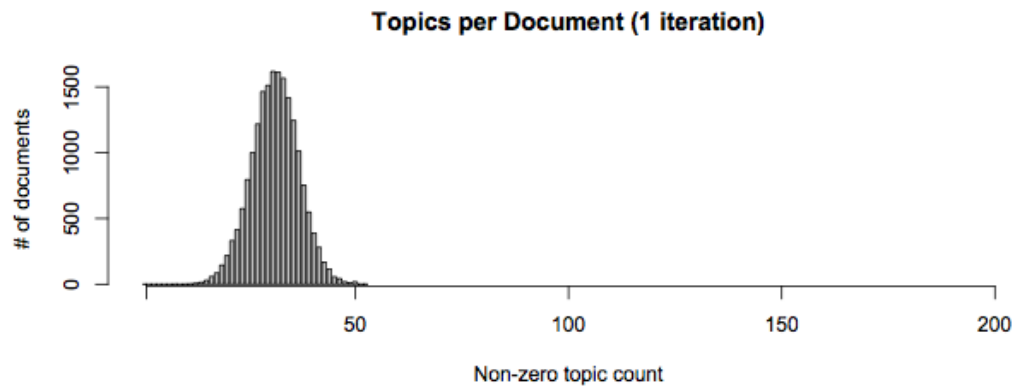
Independent of word and document

Document-specific

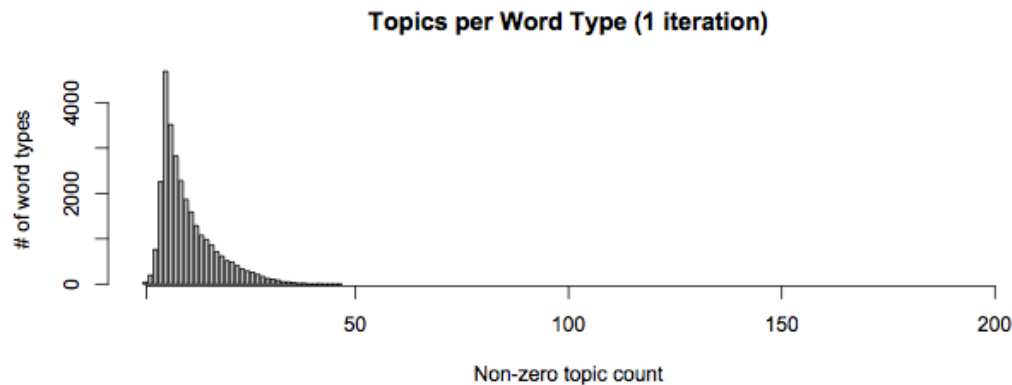
The normalizing constant

$$Z = \sum_{t: N_{w|t} > 0} \frac{N_{w|t}(N_{t|d} + \alpha)}{N_t + V\beta} + \sum_{t: N_{t|d} > 0} \frac{\beta N_{t|d}}{N_t + V\beta} + \sum_t \frac{\alpha\beta}{N_t + V\beta}$$

Statistics are sparse



- $N_{t|d}$: 10-20%
- mostly zeros



- $N_{w|t}$: < 5%
- almost all zeros

Add up the scores

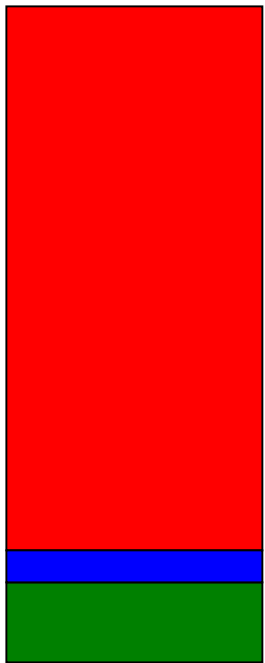
Doc A

dog	
cat	2
feline	2
hound	0



Doc B

dog	0
dog	0
horse	1
equine	1



Z

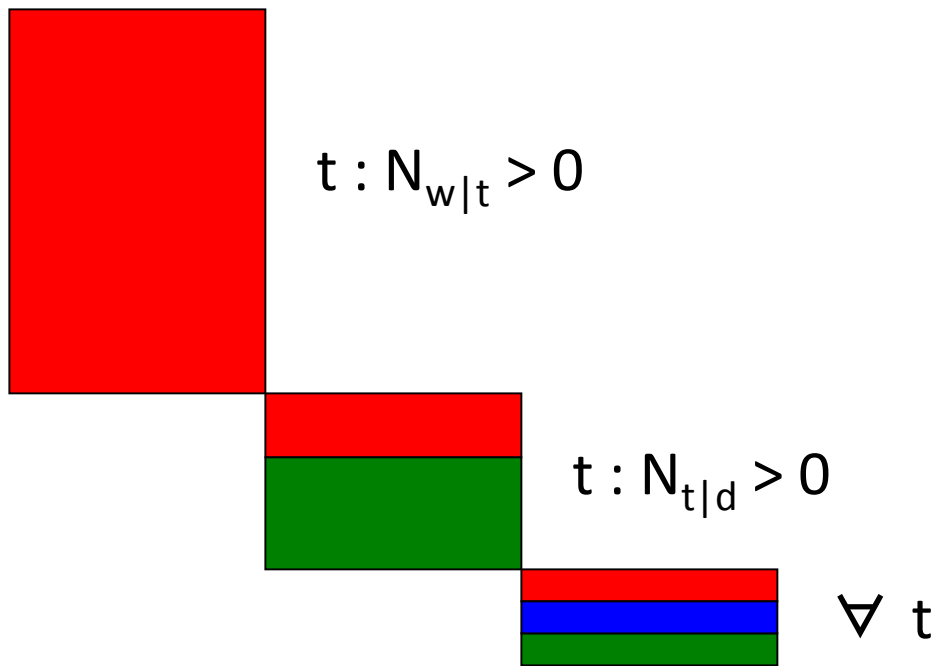
Add up the scores, in blocks

Doc A

dog	
cat	2
feline	2
hound	0

Doc B

dog	0
dog	0
horse	1
equine	1



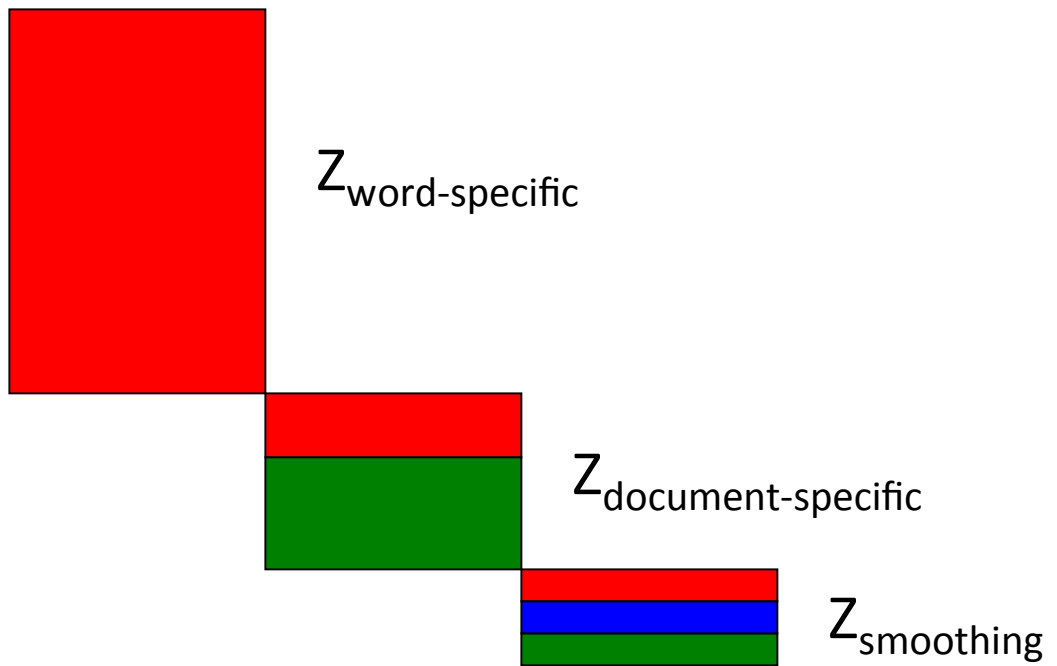
Add up the scores, in blocks

Doc A

dog	
cat	2
feline	2
hound	0

Doc B

dog	0
dog	0
horse	1
equine	1



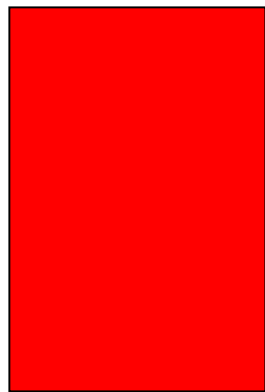
Add up *some* blocks, cache others

Doc A

dog	
cat	2
feline	2
hound	0

Doc B

dog	0
dog	0
horse	1
equine	1



Recalculate the word-specific block for every token



change *at most* two from each of these



The size of this block is *almost* constant

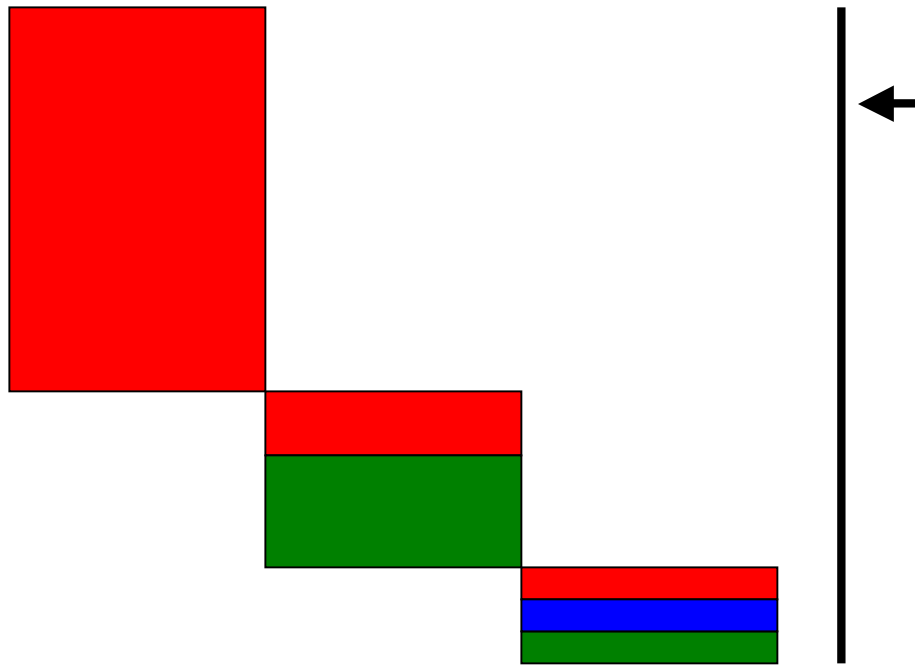
Add up *some* blocks, cache others

Doc A

dog	
cat	2
feline	2
hound	0

Doc B

dog	0
dog	0
horse	1
equine	1



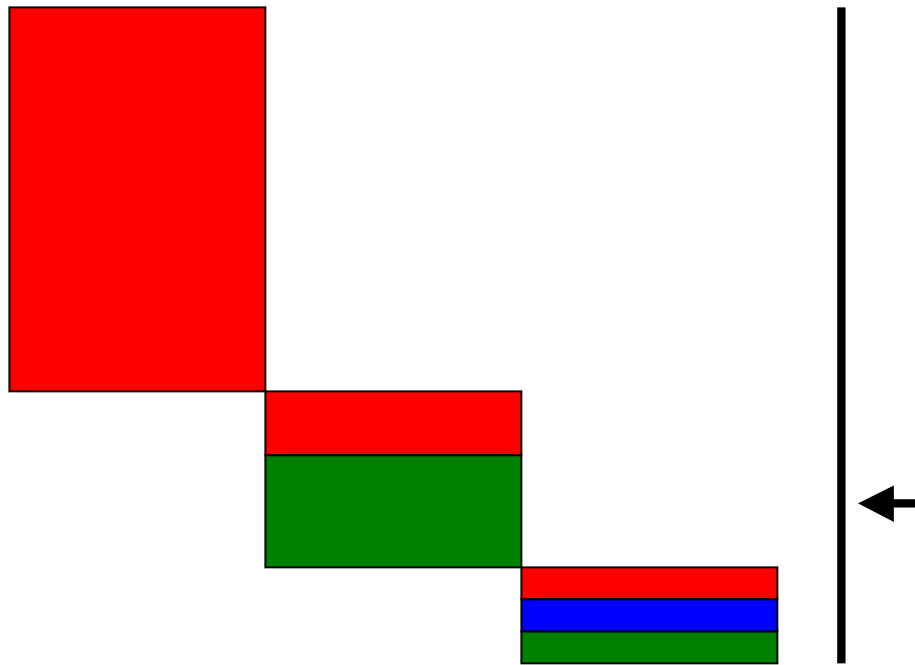
Add up *some* blocks, cache others

Doc A

dog	
cat	2
feline	2
hound	0

Doc B

dog	0
dog	0
horse	1
equine	1



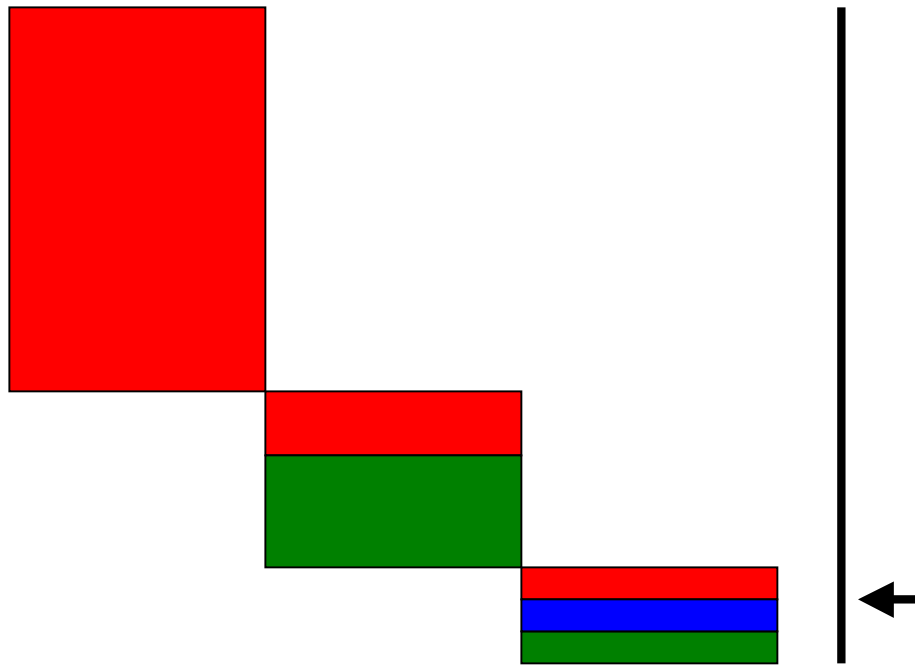
Add up *some* blocks, cache others

Doc A

dog	
cat	2
feline	2
hound	0

Doc B

dog	0
dog	0
horse	1
equine	1



Worst case: we might have to loop over all topics

Summary so far

- We can store most of the computation to calculate Z from token to token.
- We can sample exactly from the same distribution, with a new map from $(0, Z)$ to topics.

Fast iteration over $\{t : N_{w|t} > 0\}$ is *critical*

Topic-word statistics

Doc A

dog	0
cat	2
feline	2
hound	0

Doc B

dog	0
dog	0
horse	1
equine	1

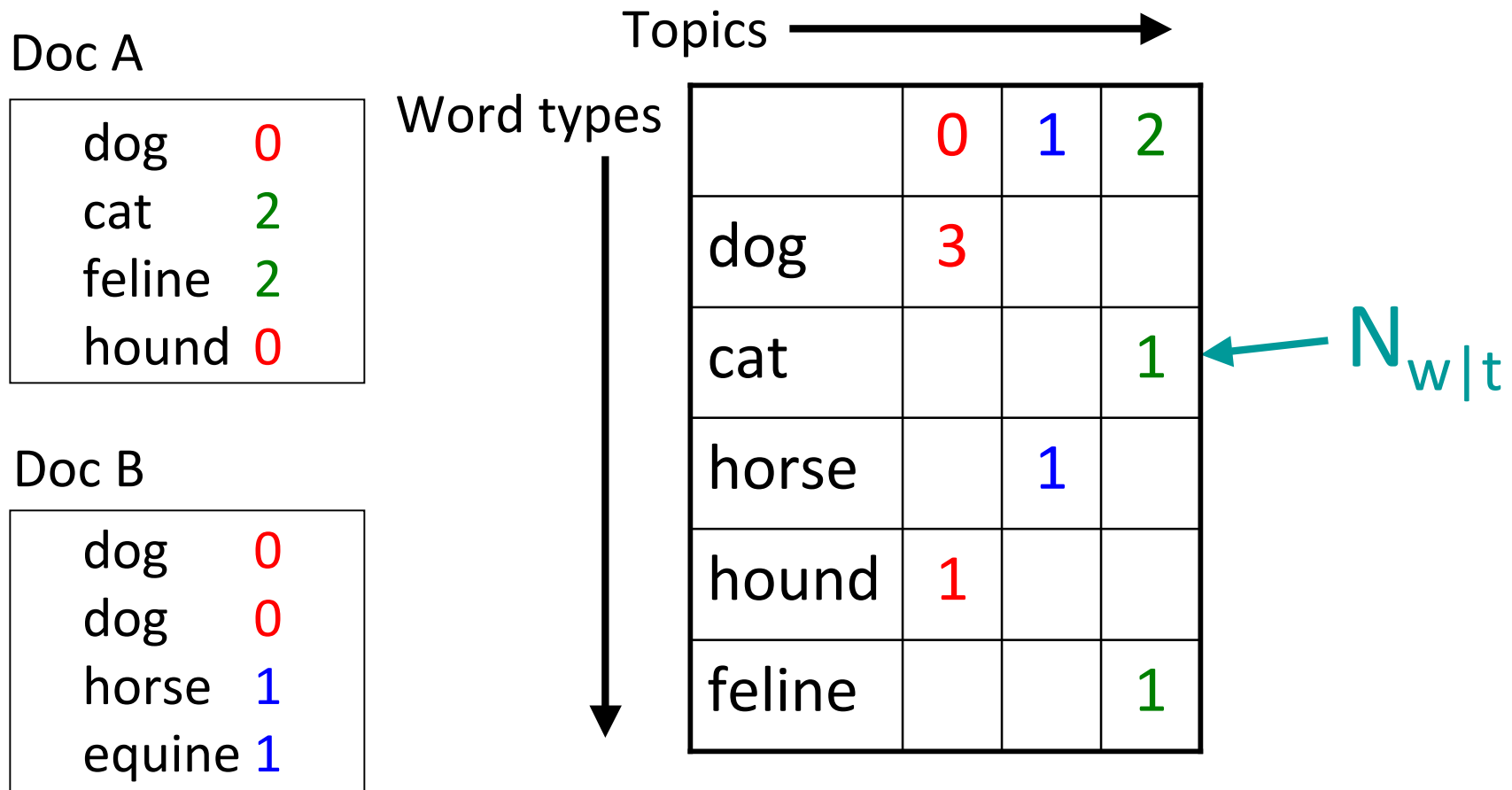
Word types \longrightarrow

Topics \downarrow

	dog	cat	horse	hound	feline
0	3			1	
1			1		
2		1			1

$N_{w|t}$

Word-topic statistics *transpose*



Representations of $N_{w|t}$: arrays

```
int[] typeTopicCounts = new int[T];
```

Topics index array positions.

0	1	2	3
3	0	0	1

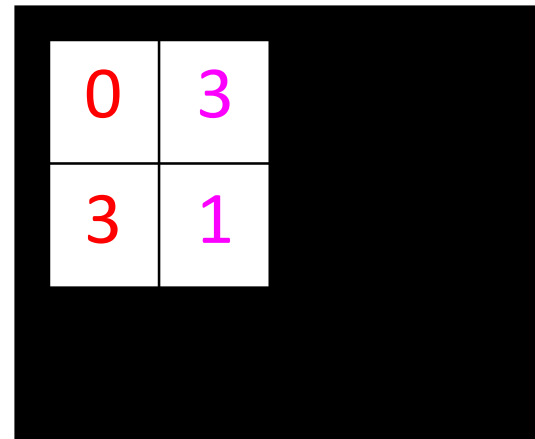
Most entries are zero.

Integer HashMaps

```
TIntIntHashMap typeTopicCounts =  
    new TIntIntHashMap();
```

Faster, but...

- Complicated
- Not much memory improvement
- Adds dependencies on external libraries (trove, fastutil).

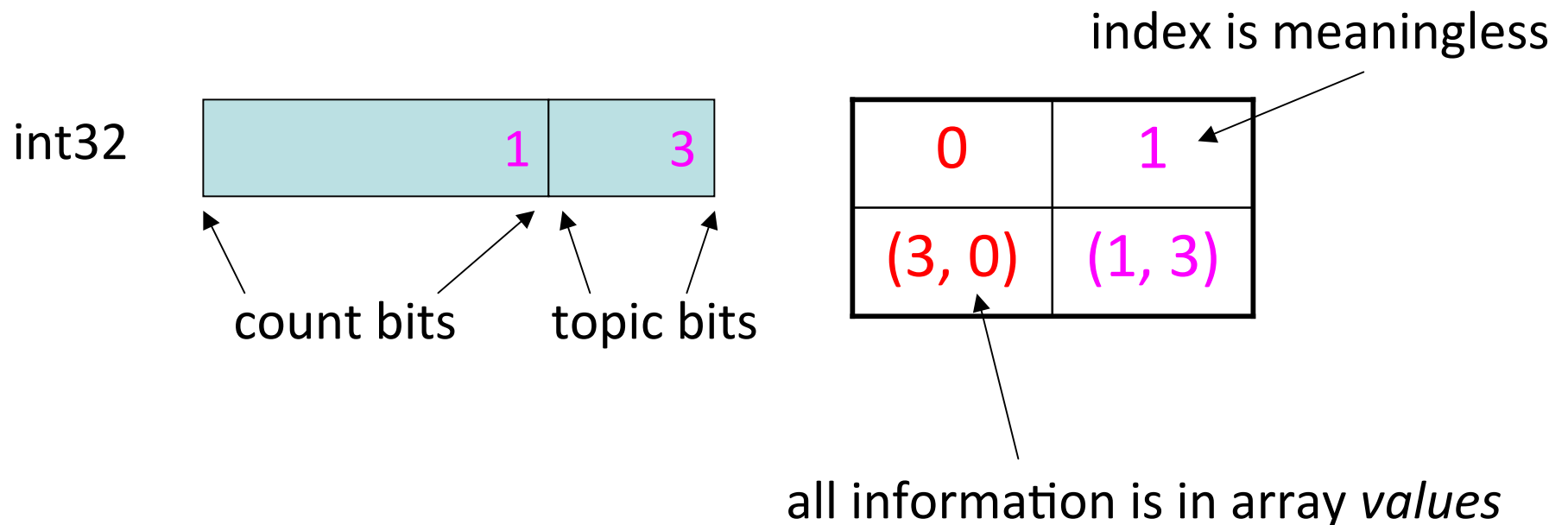


0	3
3	1

(giant black box)

Encoded integer arrays

```
int[] typeTopicCounts = new int[N_w]
```



Huge speedup, 2x faster than HashMaps

Example

	...	8	...	15	...	83	...	96	...
"dog"	0	3	0	24	0	9	0	1	0

100 topics, so 4x100 bytes = 400

Example

	...	8	...	15	...	83	...	96	...
"dog"	0	3	0	24	0	9	0	1	0

0	1	2	3
(24, 15)	(9, 83)	(3, 8)	(1, 96)

4 non-zero topics, 4 x 4 bytes = 16!

Example

	...	8	...	15	...	83	...	96	...
"dog"	0	3	0	24	0	9	0	1	0

0	1	2	3
$24 \ll 7 + 15$	$9 \ll 7 + 83$	$3 \ll 7 + 8$	$1 \ll 7 + 96$

100 topics, $2^7 > 100$

Basic Operations: iteration

0	1	2	3
(24, 15)	(9, 83)	(3, 8)	(1, 96)



```
t = v & 0x11111111
```

```
Nw|t = v >> 7
```

Basic Operations: iteration

0	1	2	3
(24, 15)	(9, 83)	(3, 8)	(1, 96)



$z_{\text{word-specific}} =$

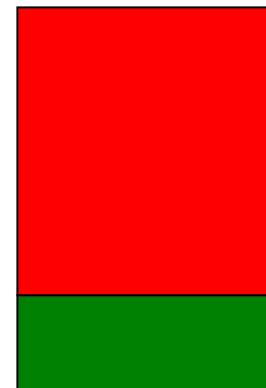


Basic Operations: iteration

0	1	2	3
(24, 15)	(9, 83)	(3, 8)	(1, 96)



$z_{\text{word-specific}} =$

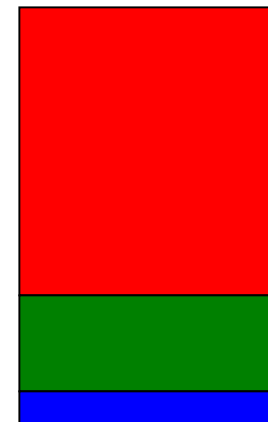


Basic Operations: iteration

0	1	2	3
(24, 15)	(9, 83)	(3, 8)	(1, 96)



$Z_{\text{word-specific}} =$



Basic Operations: increment

0	1	2	3
(24, 15)	(9, 83)	(3, 8)	(1, 96)



$N_{w|83}^{++}$

Basic Operations: increment

0	1	2	3
(24, 15)	(9, 83)	(3, 8)	(1, 96)



$$v = (9+1) \ll 7 + 83$$

$N_{w|83}^{++}$

Basic Operations: increment

0	1	2	3
(24, 15)	(10, 83)	(3, 8)	(1, 96)



$$v = (9+1) \ll 7 + 83$$

$N_{w|83}^{++}$

Basic Operations: increment

0	1	2	3
(1, 15)	(1, 83)	(1, 96)	0



$$v = (1+1) \ll 7 + 83$$

$N_{w|83}^{++}$

Basic Operations: increment

0	1	2	3
(1, 15)	(1, 83)	(1, 96)	0



$$v = (1+1) \ll 7 + 83$$

$N_{w|83}^{++}$

Basic Operations: increment

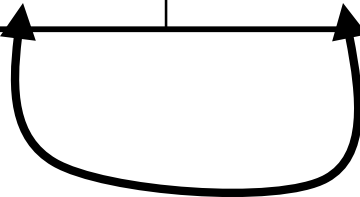
0	1	2	3
(1, 15)	(2, 83)	(1, 96)	0



$N_{w|83}^{++}$

Basic Operations: increment

0	1	2	3
(2, 83)	(1, 15)	(1, 96)	0



$N_{w|83}^{++}$

Basic Operations: decrement

0	1	2	3
(1, 15)	(1, 83)	(1, 96)	0



$$v = (1-1) \ll 7 + 83$$

$N_{w|83}^{--}$

Basic Operations: decrement

0	1	2	3
(1, 15)	0	(1, 96)	0

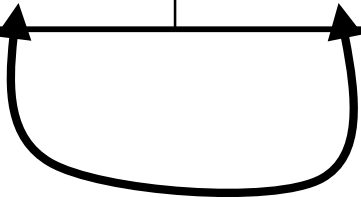


$$v = (1-1) \ll 7 + 83$$

$N_{w|83}^{--}$

Basic Operations: decrement

0	1	2	3
(1, 15)	(1, 96)	0	0



$N_{w|83}--$


Basic Operations

- Iterate:
 - linear in number of *non-zero* topics
- Increment and decrement:
 - theoretically the same as iterate, but usually constant

Memory allocation

- Conservative:
 - Count total occurrences of each word type: N_w
 - For each word type, allocate `int[$\min(T, N_w)$]`
- More savings:
 - For each type allocate `int[$\min(T, N_w)/k$]`, increase memory as needed (or not)

Conservative allocation

Topics 

“dog”



“the”

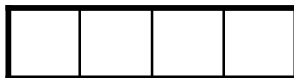


“llama”

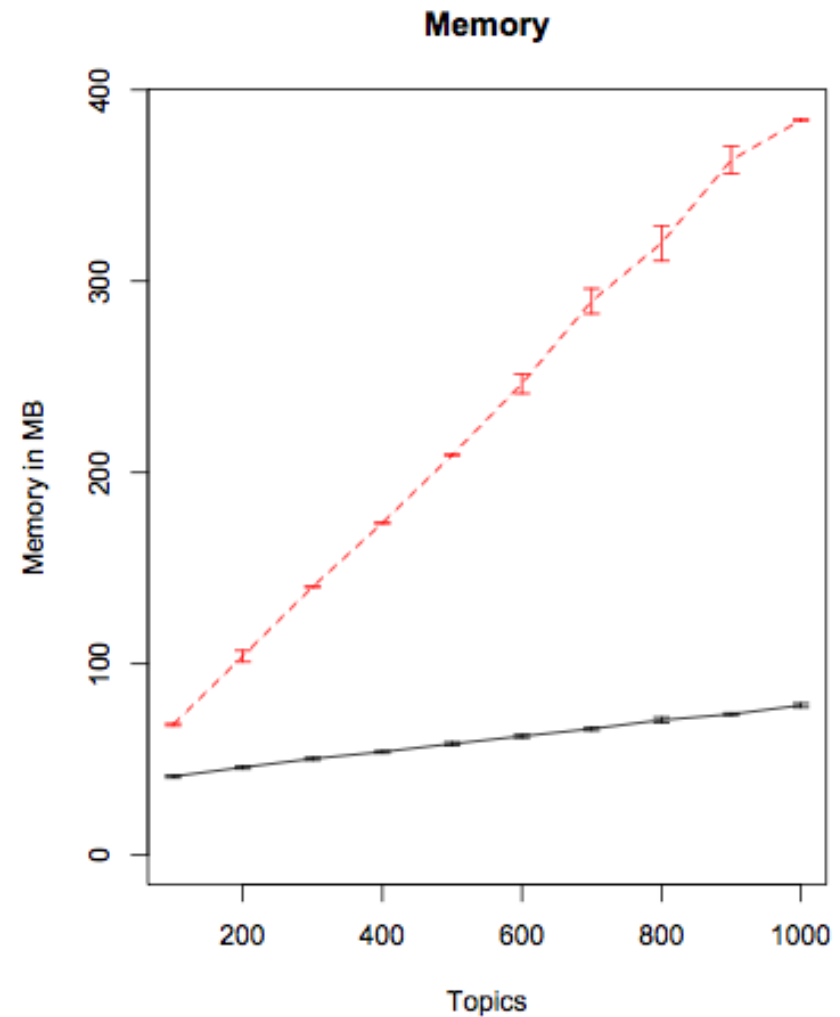
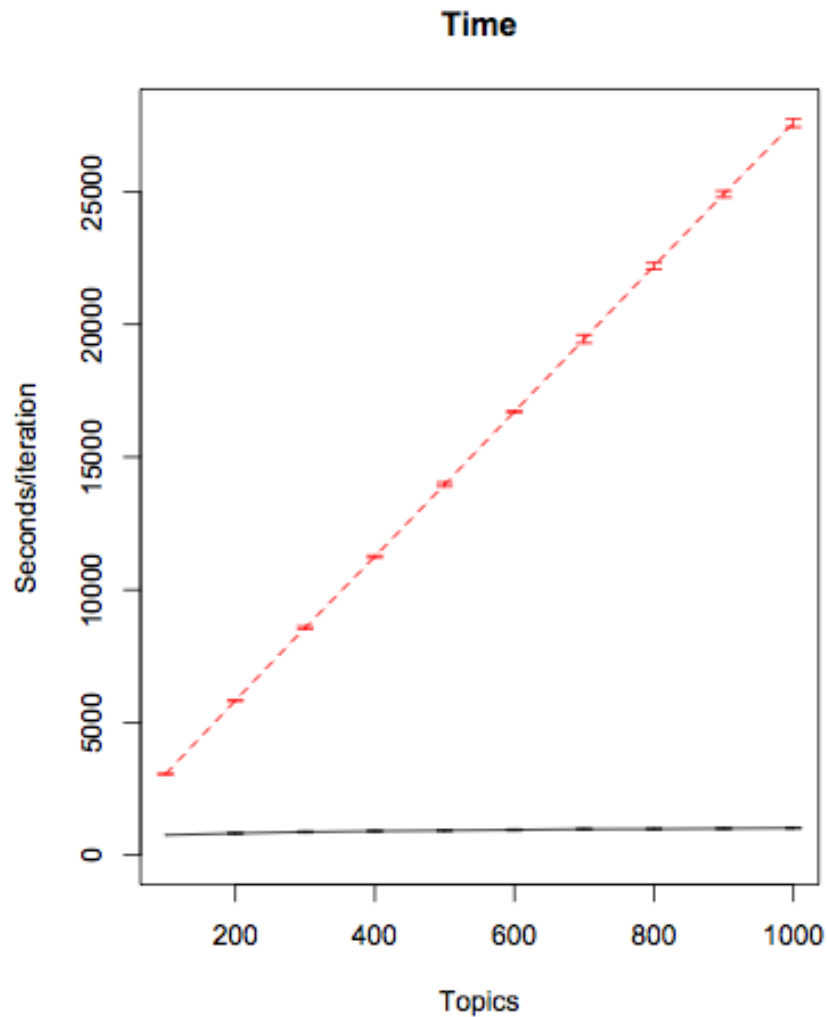


← “llama” occurs only twice in corpus, so
at most two topics

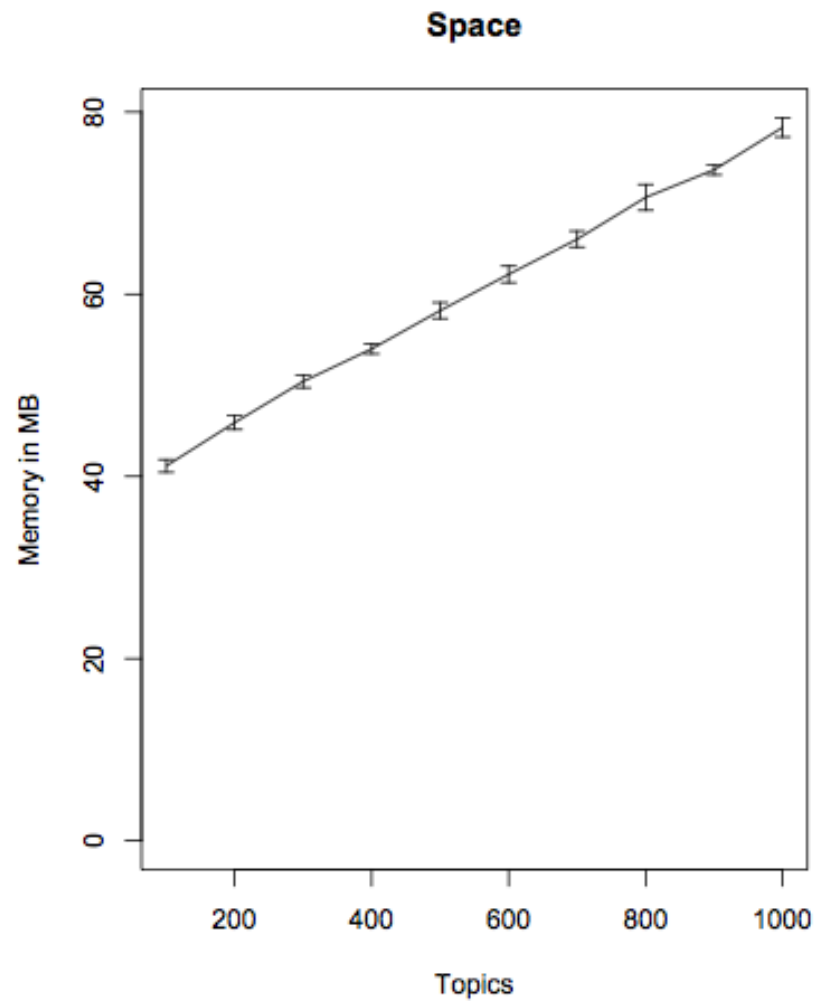
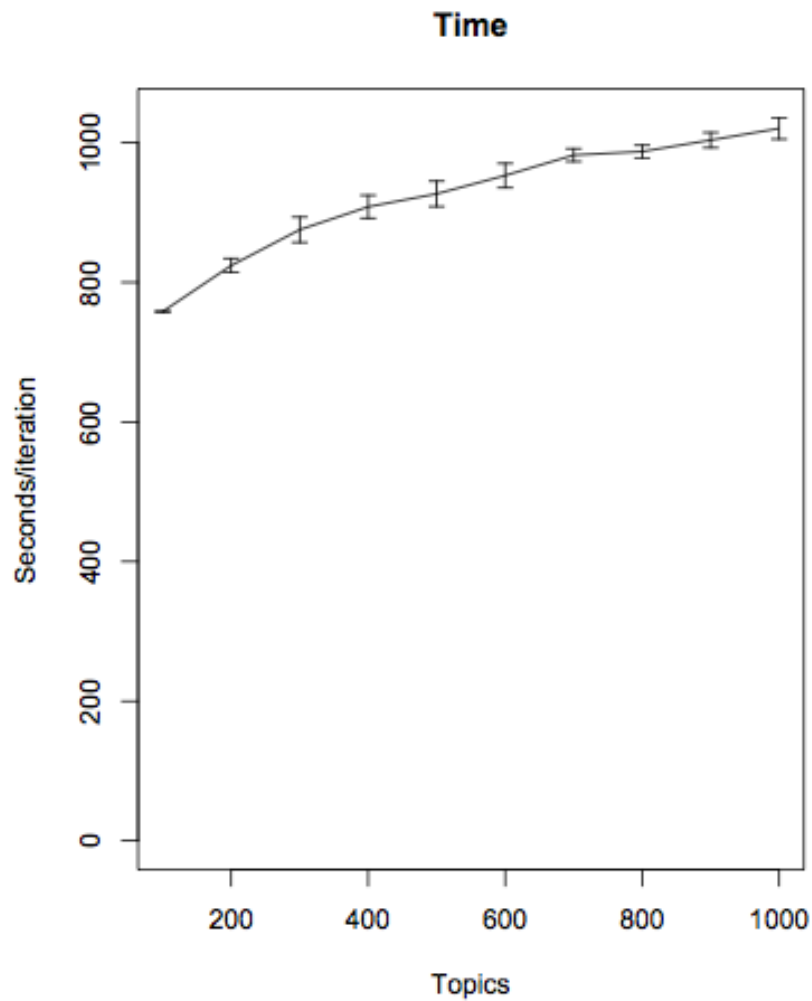
...



Huge savings: time and memory



Huge savings: time and memory



Large-scale Polylingual Topics

- Wikipedias in 40 languages (AR to ZH)
(Articles linking to same English page share $P(t|d)$)
- 1,200,000 *English* headwords
- 7,000,000 articles
- 300,000,000 tokens
- 4,000,000 distinct word types
 - Drop rare words (< 5 tokens)
 - Drop common words (> 5% of docs)
- 500 topics

One (1) CPU

Thanks!

- Paper:
Efficient Methods for Topic Model Inference on Streaming Document Collections. Limin Yao, David Mimno, Andrew McCallum. KDD 2009. (See section 3.4)
- Code:
<http://mallet.cs.umass.edu>
(Implemented in `cc.mallet.topics.WorkerRunnable`)
- 40 language Wikipedia model:
<http://christo.cs.umass.edu/wiki40>