Sketching Count Statistics for Approximate Bayesian Inference

Diana Cai and Zoe Ashwood

Modern data analysis problems

- Massive in size, possible infinite data stream
- High dimensional
- Complex models are needed for modeling

Probabilistic modeling and Bayesian inference

- Powerful for identifying interpretable, latent structure in data
- Provides a framework for making predictions about future observations
- Incorporate prior knowledge, share statistical strength across model

This project: we explore approximating the sufficient statistics in probabilistic models using a hashing-based probabilistic data structure (count-min sketch) perform Bayesian inference in two ubiquitous probabilistic models

Count-min sketch

[Cormode and Muthukrishnan, 2003]

Data stream $x_1, \ldots, x_M$ of tokens in {1,...,K}

Count-min sketch

[Cormode and Muthukrishnan, 2003]

Data stream x_1, …, x_M of tokens in {1,...,K}

N hash functions randomly generated from a *pairwise-independent* hash family

h_n: [K] -> [J], J << K, n=1,...,N

Count-min sketch

[Cormode and Muthukrishnan, 2003]

Data stream x_1, …, x_M of tokens in {1,...,K}

For a random hash h in the family,
Pr(h(x)=h(y)) at most 1/J

N hash functions randomly generated from a *pairwise-independent* hash family

h_n: [K] -> [J], J << K, n=1,...,N

Count-min sketch

[Cormode and Muthukrishnan, 2003]

Data stream x_1, …, x_M of tokens in {1,...,K}

For a random hash h in the family, Pr(h(x)=h(y)) at most 1/J

N hash functions randomly generated from a *pairwise-independent* hash family

h_n: [K] -> [J], J << K, n=1,...,N

Data structure: N x J counter array: C[n,j]

h_1					
h_2					
h_3					

Count-min sketch

[Cormode and Muthukrishnan, 2003]

Data stream x_1, …, x_M of tokens in {1,...,K}

For a random hash h in the family, Pr(h(x)=h(y)) at most 1/J

N hash functions randomly generated from a *pairwise-independent* hash family

h_n: [K] -> [J], J << K, n=1,...,N

Data structure: N x J counter array: C[n,j]

Update(C, k): hash k and update the counts

C[n,h_n(k)] += 1, for all n=1,...,N

h_1					
h_2					
h_3					

Count-min sketch

[Cormode and Muthukrishnan, 2003]

Data stream $x_1, \ldots, x_M$ of tokens in {1,...,K}

For a random hash h in the family, Pr(h(x)=h(y)) at most 1/J

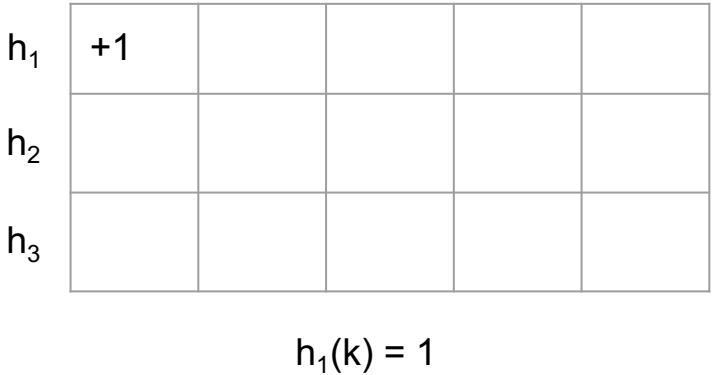
N hash functions randomly generated from a *pairwise-independent* hash family

h_n: [K] -> [J], J << K, n=1,...,N

Data structure: N x J counter array: C[n,j]

Update(C, k): hash k and update the counts

C[n,h_n(k)] += 1, for all n=1,...,N



Count-min sketch

[Cormode and Muthukrishnan, 2003]

Data stream $x_1, \ldots, x_M$ of tokens in {1,...,K}

For a random hash h in the family, Pr(h(x)=h(y)) at most 1/J

N hash functions randomly generated from a *pairwise-independent* hash family

h_n: [K] -> [J], J << K, n=1,...,N

Data structure: N x J counter array: C[n,j]

Update(C, k): hash k and update the counts

$C[n,h_n(k)]$ += 1, for all n=1,...,N

h_1	+1				
h_2				+1	
h_3					

$h_1(k) = 1$
$h_2(k) = 4$

Count-min sketch

[Cormode and Muthukrishnan, 2003]

Data stream $x_1, \ldots, x_M$ of tokens in {1,...,K}

For a random hash h in the family, Pr(h(x)=h(y)) at most 1/J

N hash functions randomly generated from a *pairwise-independent* hash family

h_n: [K] -> [J], J << K, n=1,...,N

Data structure: N x J counter array: C[n,j]

Update(C, k): hash k and update the counts

$C[n,h_n(k)]$ += 1, for all n=1,...,N

h_1	+1				
h_2				+1	
h_3		+1			

$h_1(k) = 1$
$h_2(k) = 4$
$h_3(k) = 2$

Count-min sketch

[Cormode and Muthukrishnan, 2003]

Data stream $x_1, \ldots, x_M$ of tokens in {1,...,K}

For a random hash h in the family, Pr(h(x)=h(y)) at most 1/J

N hash functions randomly generated from a *pairwise-independent* hash family

h_n: [K] -> [J], J << K, n=1,...,N

Data structure: N x J counter array: C[n,j]

Update(C, k): hash k and update the counts

C[n,h_n(k)] += 1, for all n=1,...,N

Query(C, k): returns estimate for the count of k

For all n, return the minimum count C[n,h_n(k)]

h_1	+1				
h_2				+1	
h_3		+1			

$h_1(k) = 1$
$h_2(k) = 4$
$h_3(k) = 2$

Count-min sketch

[Cormode and Muthukrishnan, 2003]

Data stream $x_1, \ldots, x_M$ of tokens in {1,...,K}

For a random hash h in the family, Pr(h(x)=h(y)) at most 1/J

N hash functions randomly generated from a *pairwise-independent* hash family

h_n: [K] -> [J], J << K, n=1,...,N

Data structure: N x J counter array: C[n,j]

Update(C, k): hash k and update the counts

$C[n,h_n(k)]$ += 1, for all n=1,...,N

Query(C, k): returns estimate for the count of k

For all n, return the minimum count $C[n,h_n(k)]$

sketch after M observations:

h_1	10				
h_2				3	
h_3		5			

$h_1(k) = 1$
$h_2(k) = 4$
$h_3(k) = 2$

Count-min sketch

[Cormode and Muthukrishnan, 2003]

Data stream $x_1, \ldots, x_M$ of tokens in {1,...,K}

For a random hash h in the family, Pr(h(x)=h(y)) at most 1/J

N hash functions randomly generated from a *pairwise-independent* hash family

h_n: [K] -> [J], J << K, n=1,...,N

Data structure: N x J counter array: C[n,j]

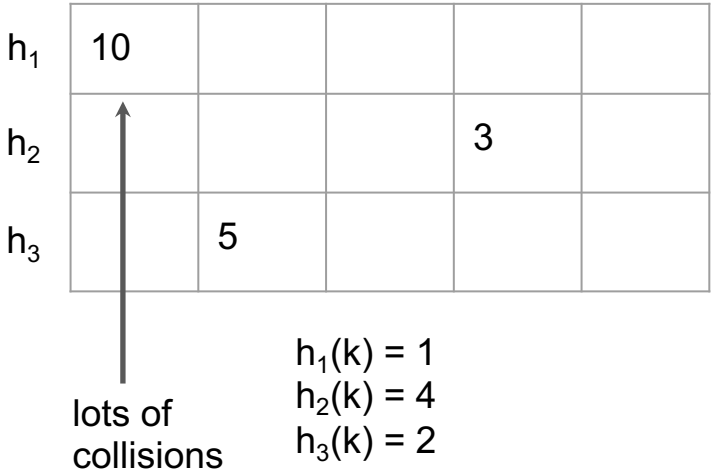
Update(C, k): hash k and update the counts

$C[n,h_n(k)]$ += 1, for all n=1,...,N

Query(C, k): returns estimate for the count of k

For all n, return the minimum count $C[n,h_n(k)]$

sketch after M observations:



Count-min sketch

[Cormode and Muthukrishnan, 2003]

Data stream $x_1, \ldots, x_M$ of tokens in {1,...,K}

For a random hash h in the family, Pr(h(x)=h(y)) at most 1/J

N hash functions randomly generated from a *pairwise-independent* hash family

h_n: [K] -> [J], J << K, n=1,...,N

Data structure: N x J counter array: C[n,j]

Update(C, k): hash k and update the counts

$C[n,h_n(k)]$ += 1, for all n=1,...,N

Query(C, k): returns estimate for the count of k

For all n, return the minimum count $C[n,h_n(k)]$

sketch after M observations:

h_1	10				
h_2				**3**	
h_3		5			

$h_1(k) = 1$
$h_2(k) = 4$
$h_3(k) = 2$

Count-min sketch analysis

[Cormode and Muthukrishnan, 2003]

1. **The estimated count ≥ true count**

observed count = true count + counts of colliding tokens

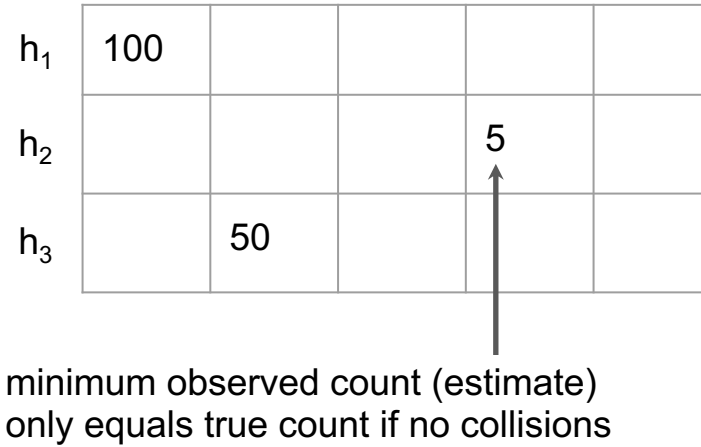
Count-min sketch analysis

[Cormode and Muthukrishnan, 2003]

1. **The estimated count ≥ true count**

   observed count = true count + counts of colliding tokens

   ⇒ estimated count := minimum observed count ≥ true count



Count-min sketch analysis

[Cormode and Muthukrishnan, 2003]

1. **The estimated count ≥ true count**

   observed count = true count + counts of colliding toker

   ⇒ estimated count := minimum observed count ≥ true count

N: # of hash functions
J: range of hash function
M: total # of tokens

1. With probability 1 - exp(-N),

   estimated count ≤ true count + (e/J) * M

Count-min sketch analysis

[Cormode and Muthukrishnan, 2003]

1. **The estimated count ≥ true count**

   observed count = true count + counts of colliding toker

   ⇒ estimated count := minimum observed count ≥ true count

N: # of hash functions
J: range of hash function
M: total # of tokens

1. With probability 1 - exp(-N),

   estimated count ≤ true count + (e/J) * M [Homework 1, Q2!]

   Proof sketch:

   X_{kn} := counts tokens colliding with k on h_n = observed - true

   $E[X_{kn}] \leq (1/J) * M$ (expected colliding mass)

   $Pr(\text{estimate} > \text{true} + (e/J) * M) = Pr(\forall n, \{X_{kn} > e * E[X_{kn}]\}) \leq \exp(-N)$

Markov's inequality

Dirichlet-multinomial (DM) model

Multinomial occupancy scheme: K bins, M balls thrown independently in each bin
Probability θ_k of landing into the k^{th} bin

Let $x_1,...,x_M \sim$ **Multinomial(θ)**. (M tokens in [K])

Under the DM model, we assume that the probabilities are *random*: $\theta \sim$ **Dirichlet$_K$(α)**

Goal is to compute the posterior distribution using Bayes' rule: $\theta \in \Delta^K$

$$p(\theta \mid x_1,...,x_M) = p(\theta)\, p(x_1,...,x_M \mid \theta) = \textbf{Dirichlet}(\theta \mid \alpha + \mathbf{c}), \quad \text{where } c = (c_1,\ldots,c_k)$$

Predictive distribution: $p(x_{M+1} = k \mid x_1,...,x_M) = \int p(x_{M+1})\, p(\theta \mid x_1,...,x_M)\, d\theta$

Sketching multinomial counts

For the DM model, we just have to keep track of the posterior through the sufficient statistic vector c, i.e., the counts of the tokens.

In a streaming setting, we can directly apply the CMsketch for inference:

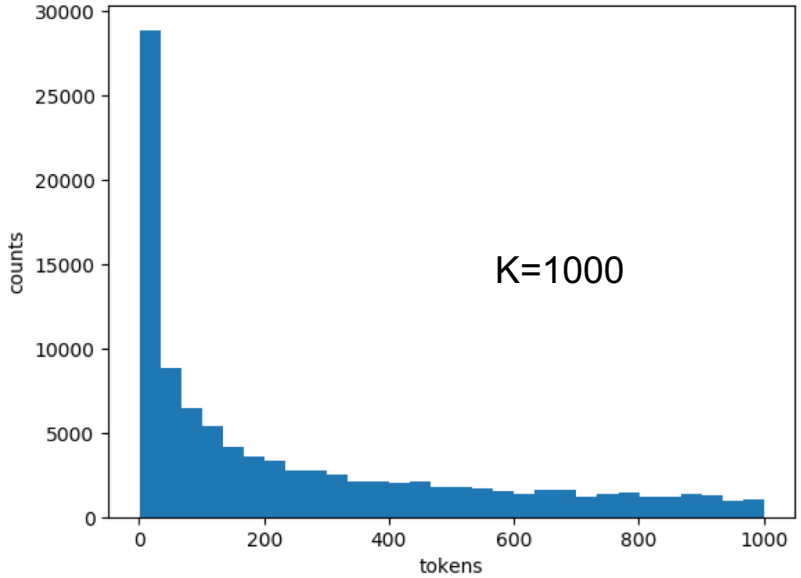
for t = 1, 2, ...

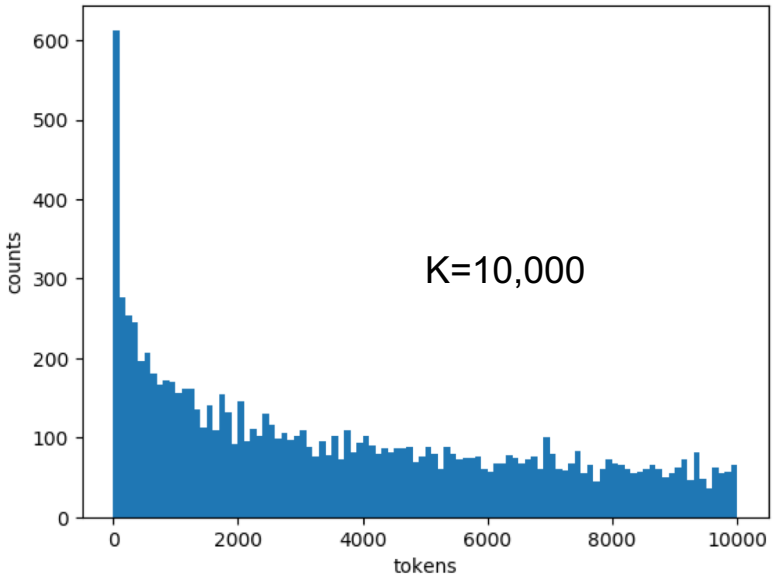
1. Observe x_t
2. Update(C, x_t): (hash x_t and update all the counters)
3. Evaluate predictive distribution

$$p(x_{M+1}= k \mid x_1,...,x_M) = (\alpha + \hat{c}_k) (\sum_j \alpha_j + \hat{c}_j)^{-1}$$

CM sketch simulations on multinomial tokens

Generated 10K size dataset from the Dirichlet-multinomial model





Applied the CMsketch (N hash functions with range J) to approximate the counts

CM sketch simulations on multinomial tokens

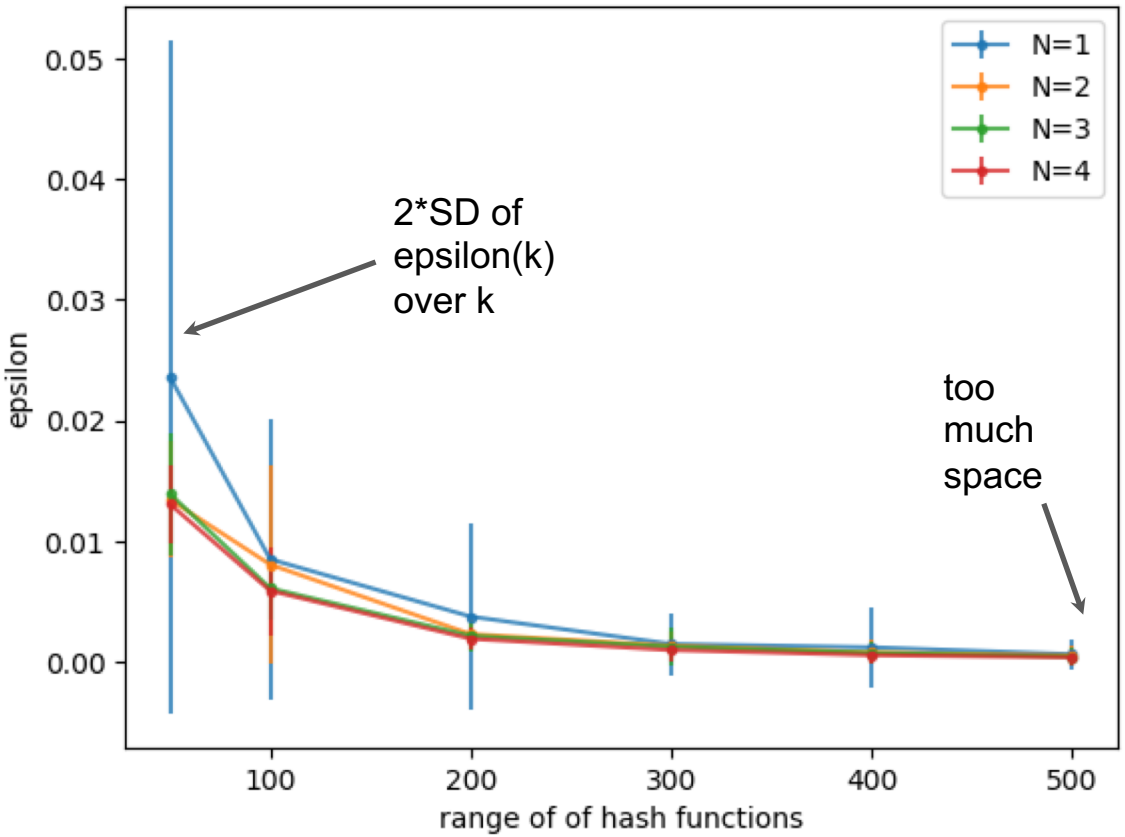
M=10,000; K=1000
N hash functions, J range

epsilon(k) := ($\hat{c}_k$- c_k) / M

plotted mean over all k

1 hash function: errors are much worse, more variance

For 1 trial:



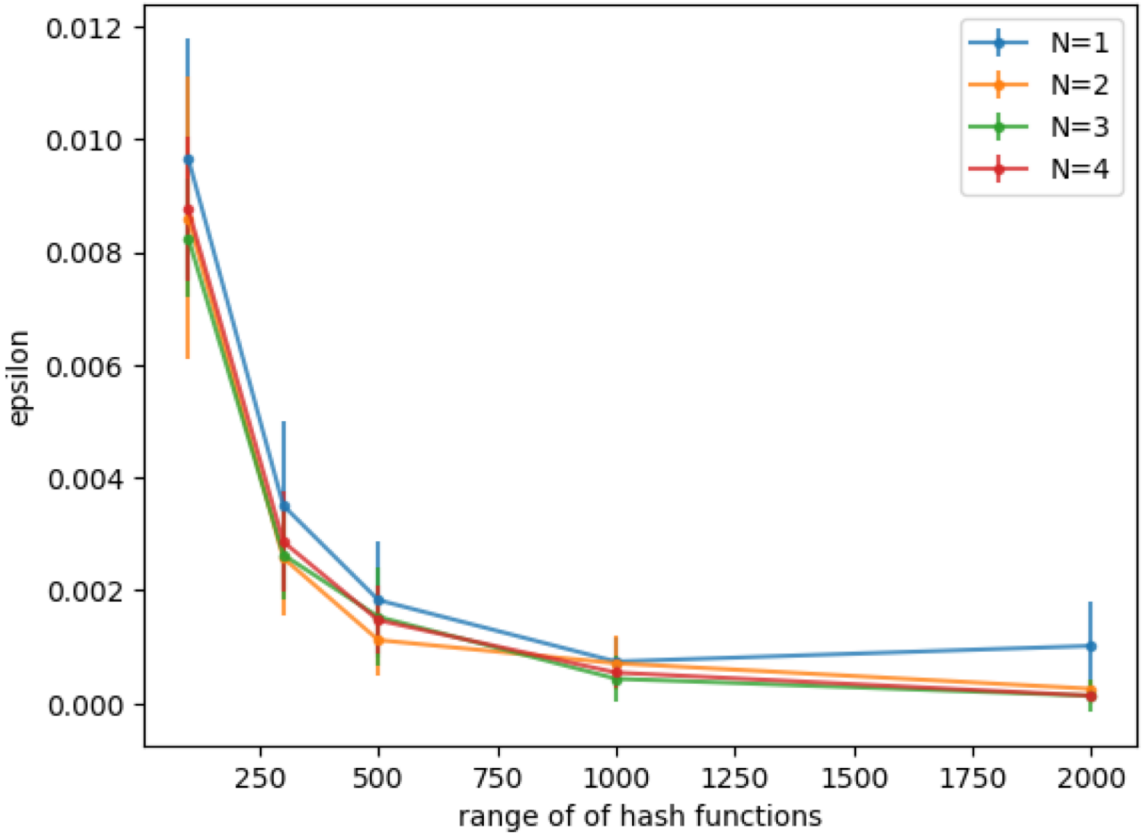
CM sketch simulations on multinomial tokens

M=10,000; K=10,000
N hash functions, J range

epsilon(k) := $(\hat{c}_k - c_k) / M$

plotted mean over all k

For 1 trial:

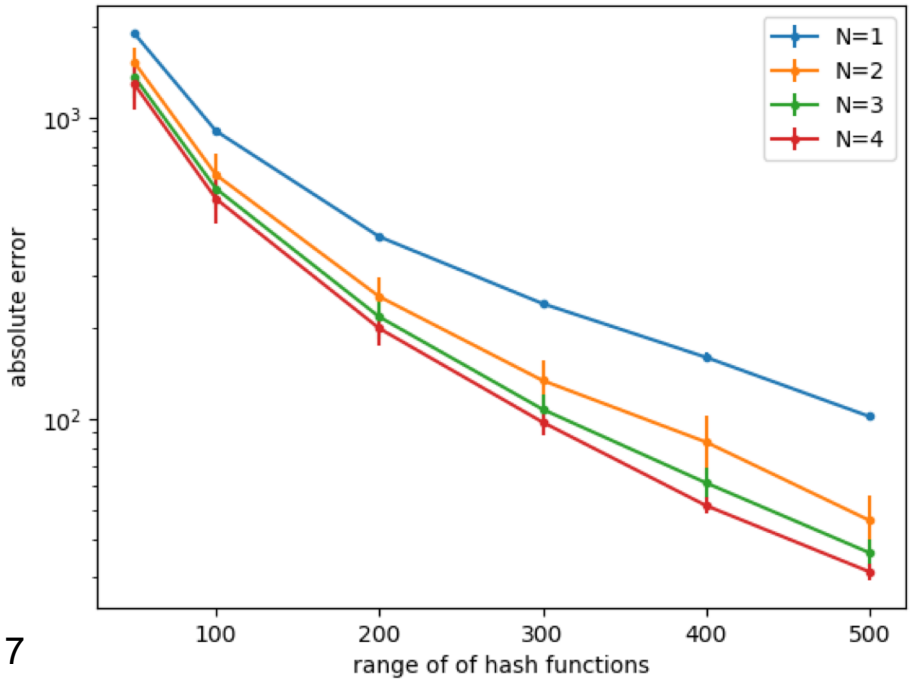


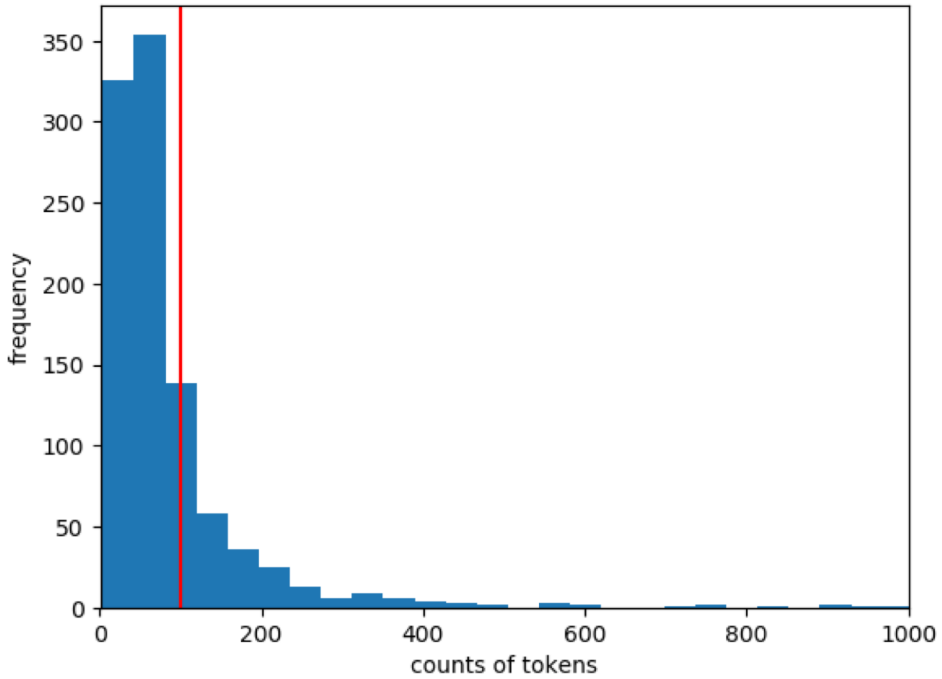
CM sketch simulations on multinomial tokens

M=10,000; K=1000; N hash functions, J range

$$MAE(\hat{c},c) = \text{sum}(\hat{c}_k - c_k) / K$$

For 20 trials: (mean over trials, sd computed over trials)



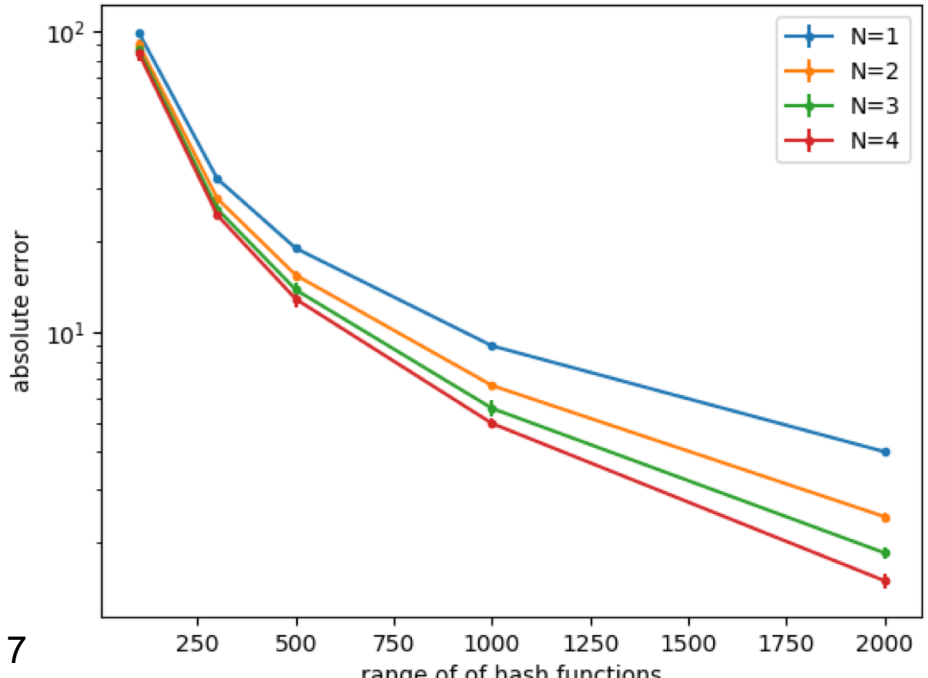


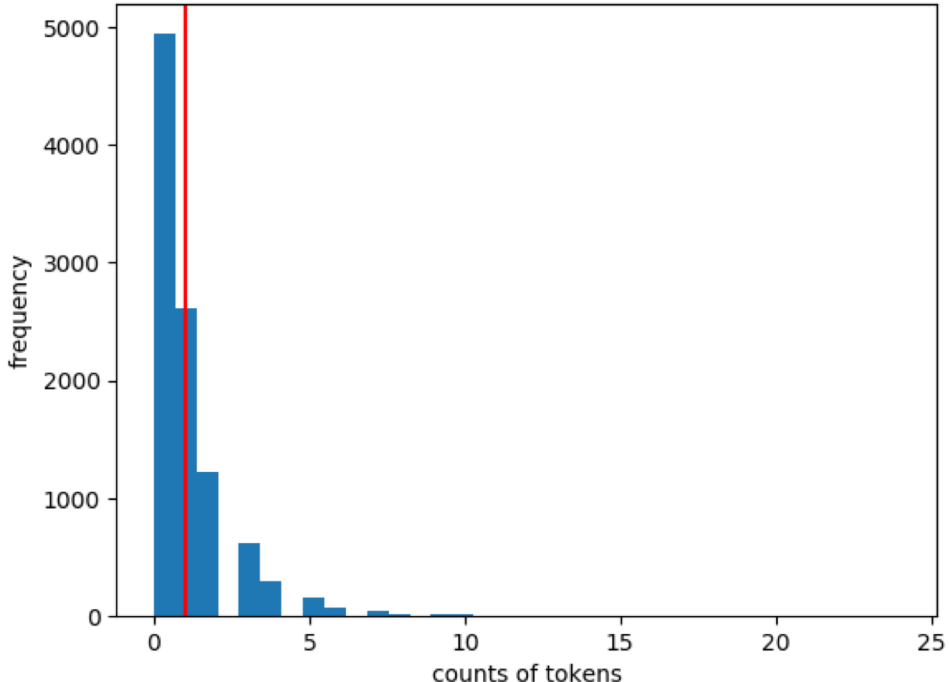
CM sketch simulations on multinomial tokens

M=10,000; K=10,000; N hash functions, J range

$$MAE(\hat{c},c) = \text{sum}(\hat{c}_k - c_k) / K$$

For 20 trials: (mean over trials, sd computed over trials)

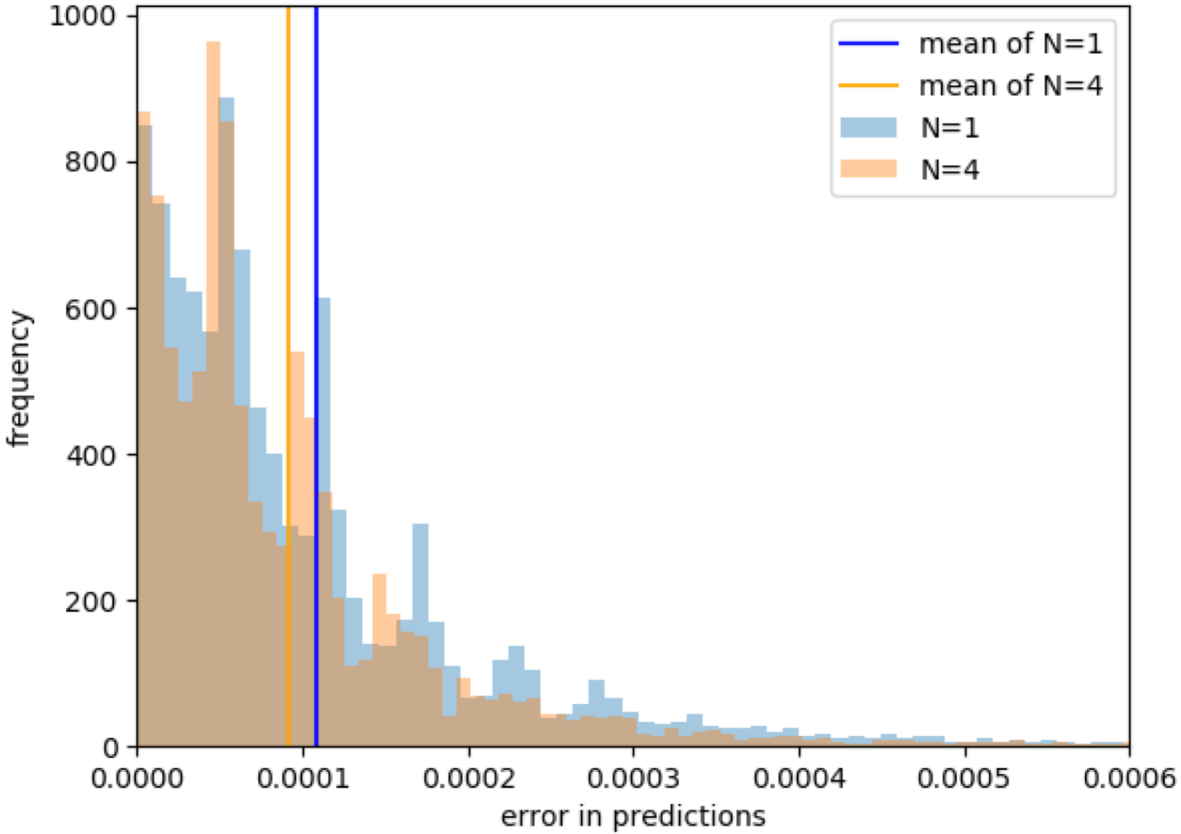




CM sketch simulations on multinomial tokens

M=10,000; K=10,000;
N hash functions, J=1000

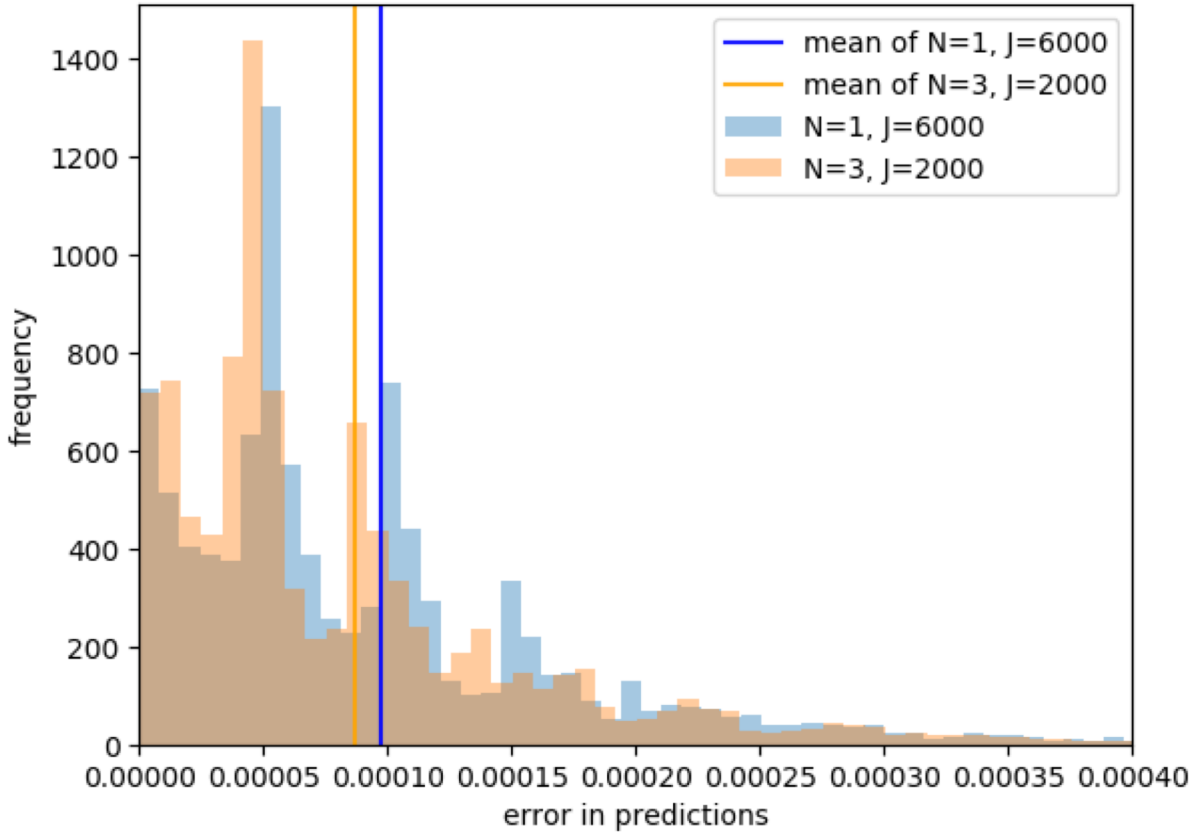
Posterior approximation:
1-step ahead
predictive probabilities
(streaming)



CM sketch simulations on multinomial tokens

M=10,000; K=10,000;
N hash functions

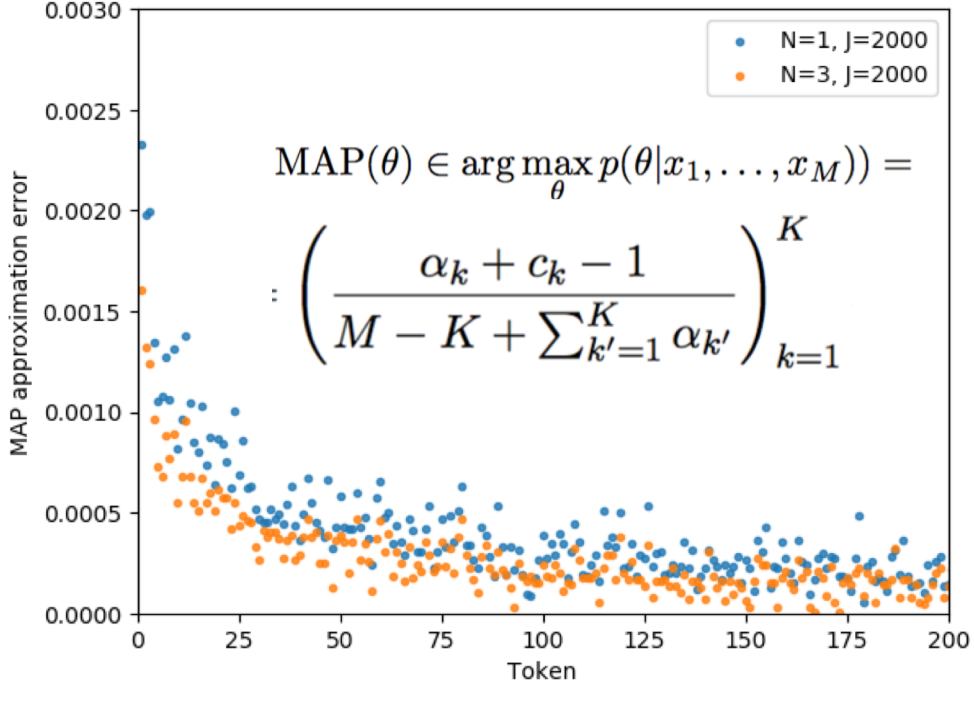
Posterior approximation:
1-step ahead
predictive probabilities
(streaming)

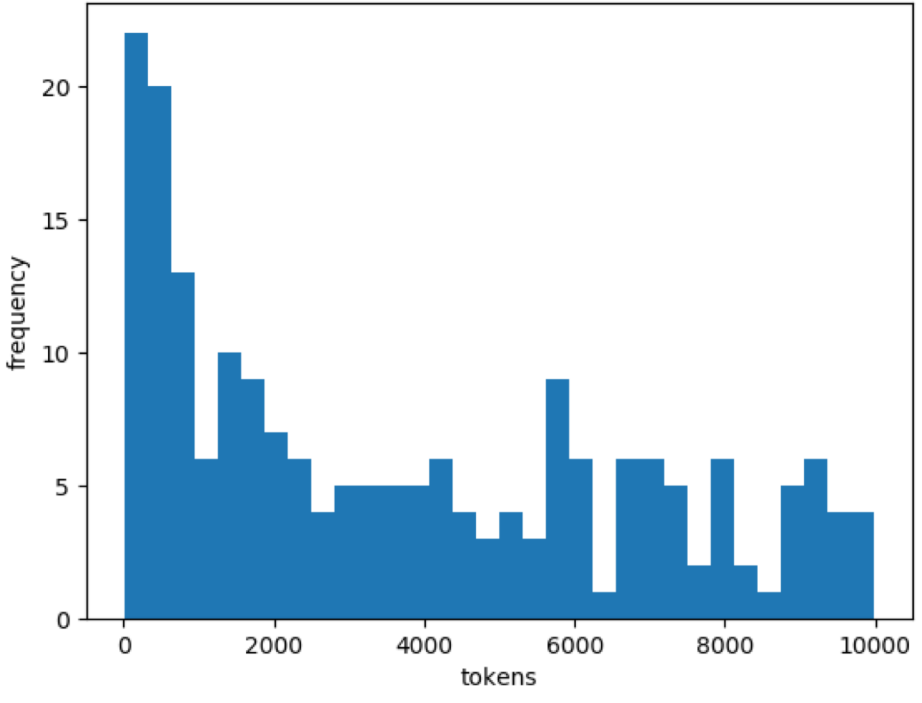


CM sketch simulations on multinomial tokens

M=10,000; K=10,000; N hash functions, J range

Posterior approximation: MAP estimate (the mode of posterior distribution)

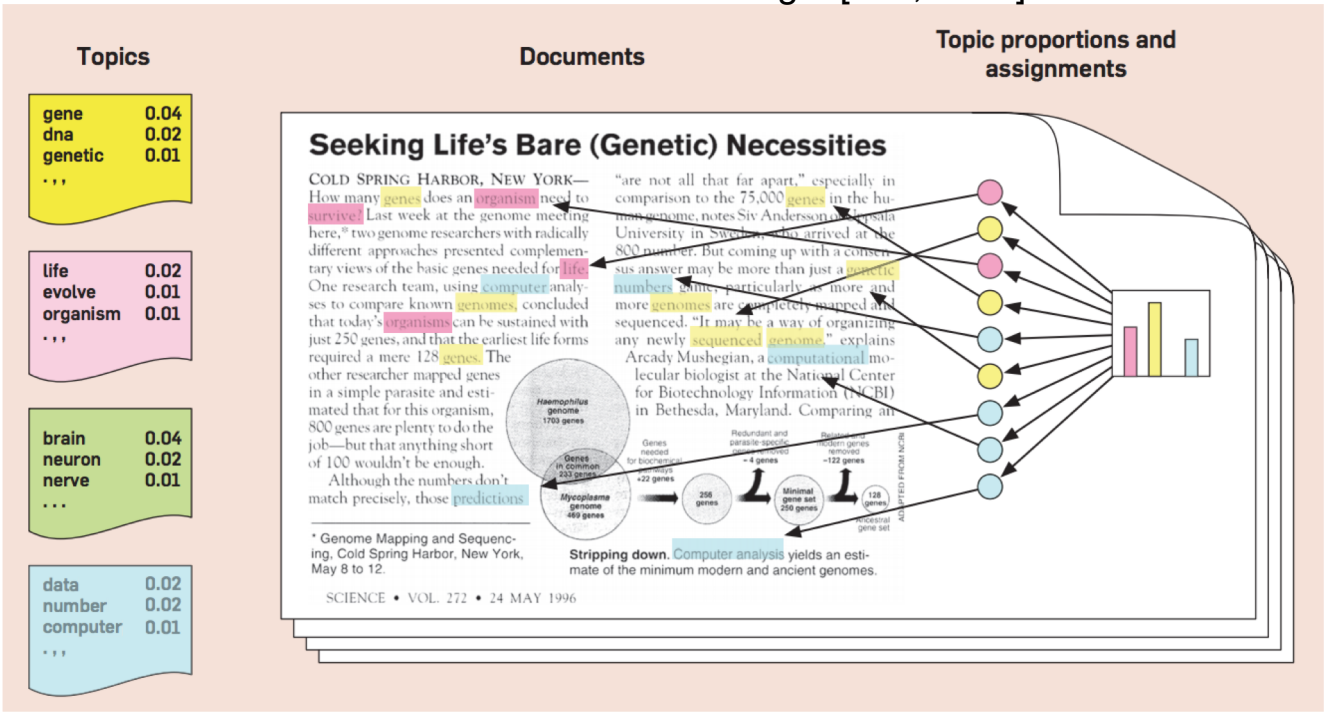




Topic modeling with Latent Dirichlet Allocation (LDA)

- Document corpus: D documents, over vocabulary set [N]
- Topic: probability distribution over words in the vocabulary set
- Goal: uncover K topics in corpus

Image: [Blei, 2012]



Topic modeling with Latent Dirichlet Allocation (LDA)

- K topics, D documents (each of length n words)
- Each topic is a distribution over the words in the vocabulary set [N]

$$\text{Draw topic } \beta_k \sim \text{Dirichlet}_N(.), \quad k = 1,...,K, \quad (\beta_k \in \Delta^N)$$

- Each document contains some proportion of the topics

$$\text{Draw } \theta_d \sim \text{Dirichlet}_K(.), \quad d = 1,..., D, \quad (\theta_d \in \Delta^K)$$

- For the j^{th} word in document d, assign that word to a topic

$$\text{Draw } z_{dj} \sim \text{Multinomial}(\theta_d), \quad j = 1, \ldots, n, \quad (z_{dj} \in [K])$$

- Each word in a document is then drawn from that topic

$$\text{Draw } w_{dj} \sim \text{Multinomial}(\beta_i), \text{ where } i = z_{dj}, \quad j = 1,...n, \quad (w_{dj} \in [N])$$

Gibbs sampling with approximate count statistics

- We want to know the posterior:

$$p(z|w) = \frac{p(z,w)}{\sum_z p(z,w)}$$

- We can write analytic form using model assumptions
- But cannot evaluate on any real corpus: sum of K^n terms in denominator is intractable (n is number of words in entire corpus)
- Must use an approximation method: can use Gibbs sampling
- Iterative algorithm that is guaranteed to converge to the true posterior

Gibbs sampling with approximate count statistics

- T iterations; each iteration involves sampling a new topic assignment for each word in the corpus
- The distribution that the topic assignment is sampled from is constantly being updated
- 5 data structures involved: one is $n_{k,\,w}$, a matrix in R^{KxN}, containing the number of times word w appears in topic k. Problem: too large for large vocabularies, N.
- Core idea: can we approximate $n_{k,\,w}$ with K count-min sketches? How is the posterior distribution, $p(z|w) = \frac{p(z,w)}{\sum_z p(z,w)}$ affected? Can we still recover the topics in a corpus?

Topic-word probabilities and sketching

- With probability $1 - \frac{1}{e^H}$, counts are in range:

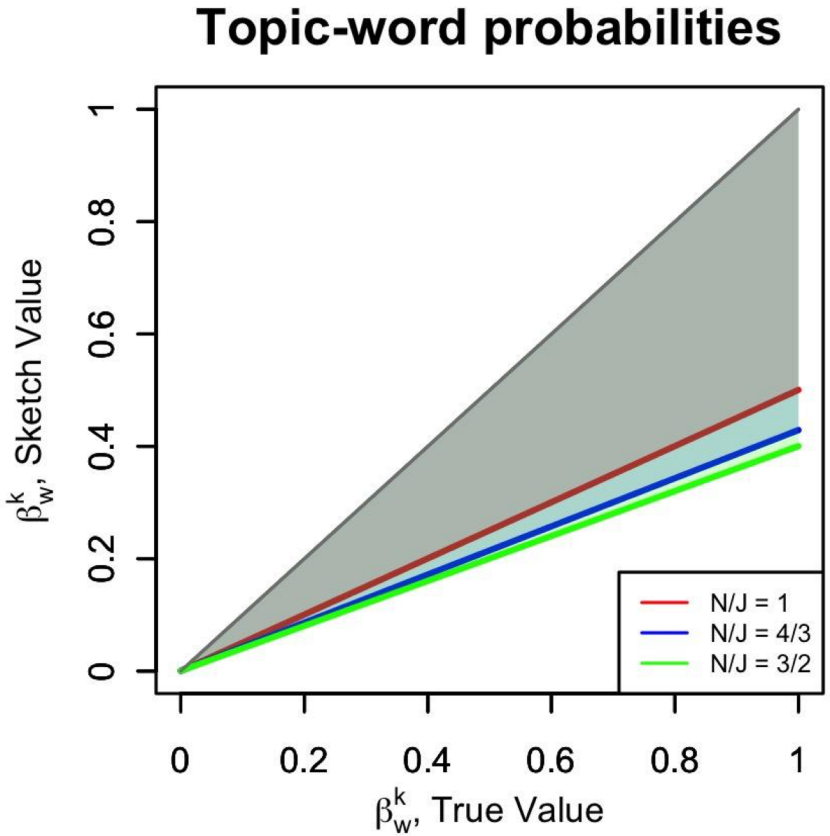
$$n_{k,w}^{\text{true}} \leq n_{k,w}^{\text{sketch}} \leq n_{k,w}^{\text{true}} + \frac{1}{J} \sum_w n_{k,w}^{\text{true}}$$

- Now: we want to estimate topic-word probabilities, $\beta_w^{k,\text{true}} \equiv \frac{n_{k,w}^{\text{true}} + \eta}{\sum_w n_{k,w}^{\text{true}} + N\eta}$

- With same high probability, these are in range:

$$\frac{1}{(1 + \frac{N}{J})}(\beta_w^{k,\text{true}} + \frac{1}{J}) \leq \beta_w^{k,\text{sketch}} \leq \beta_w^{k,\text{true}}$$

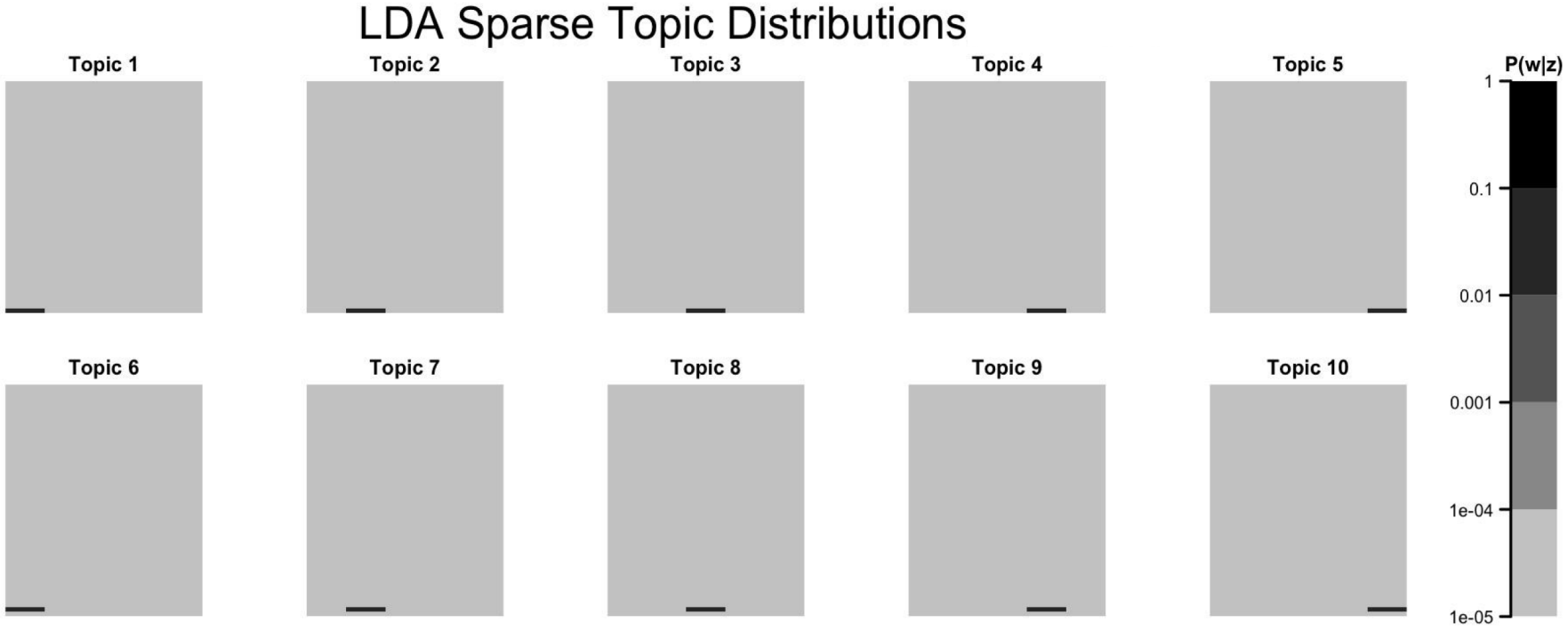
Topic-word probabilities and sketching



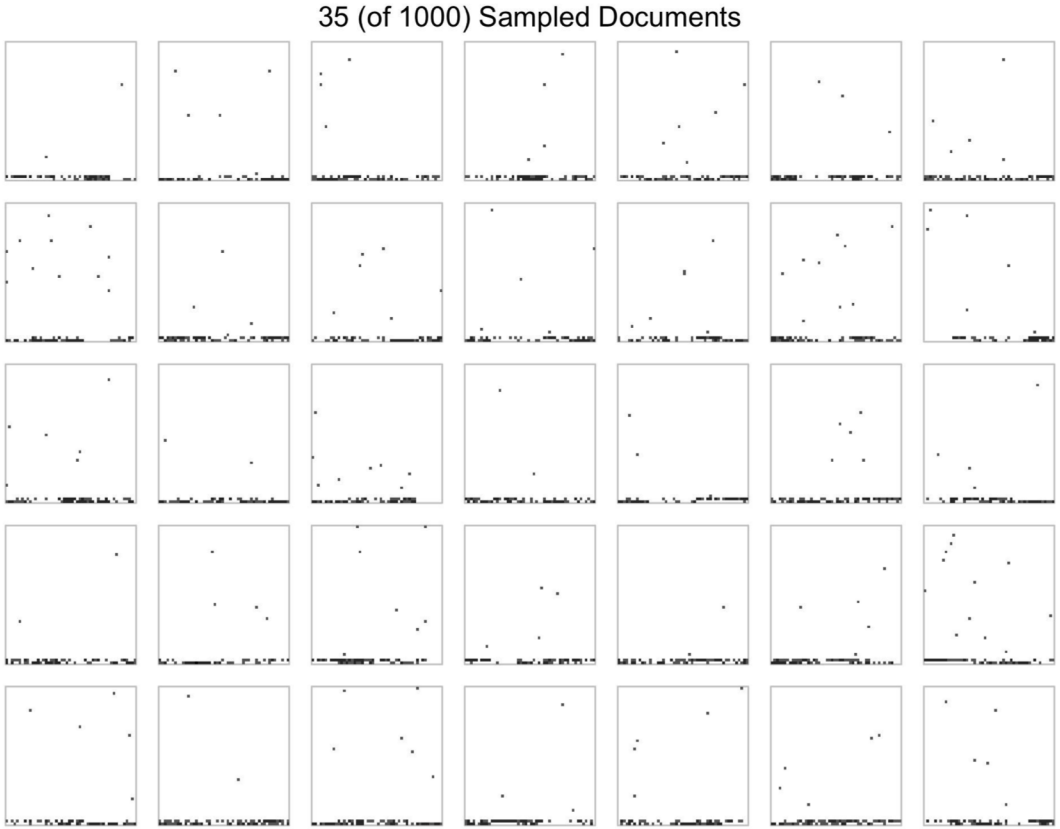
N = 2500
(vocabulary size)

J: range of hash functions

LDA Toy Corpus: 100,000 words; N = 2500

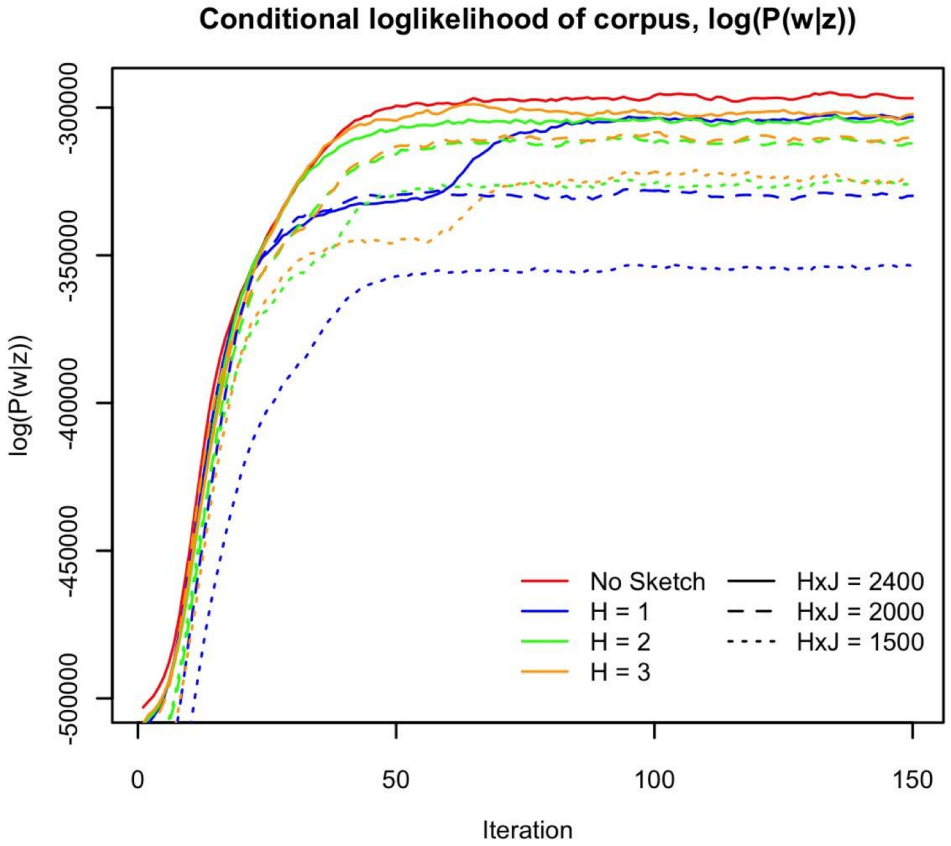


LDA Toy Corpus: 100,000 words; N = 2500

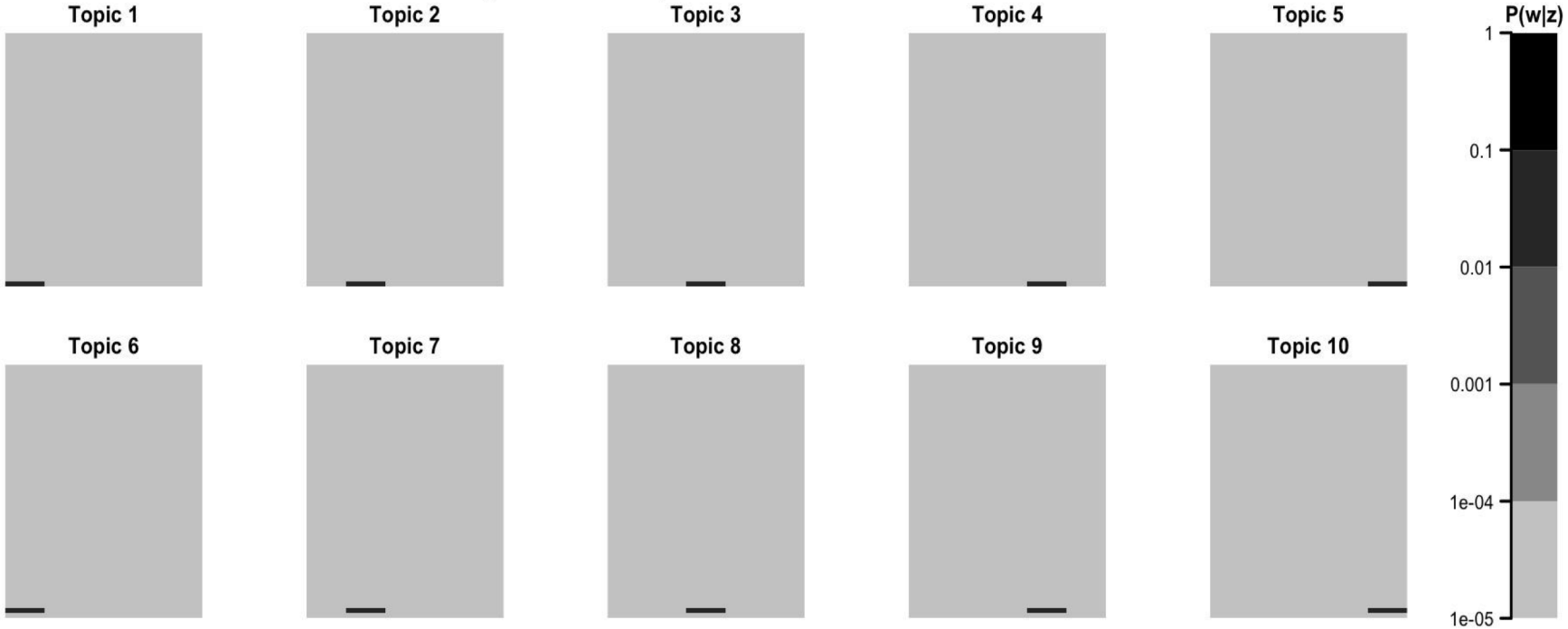


Results: Conditional Loglikelihood of Corpus

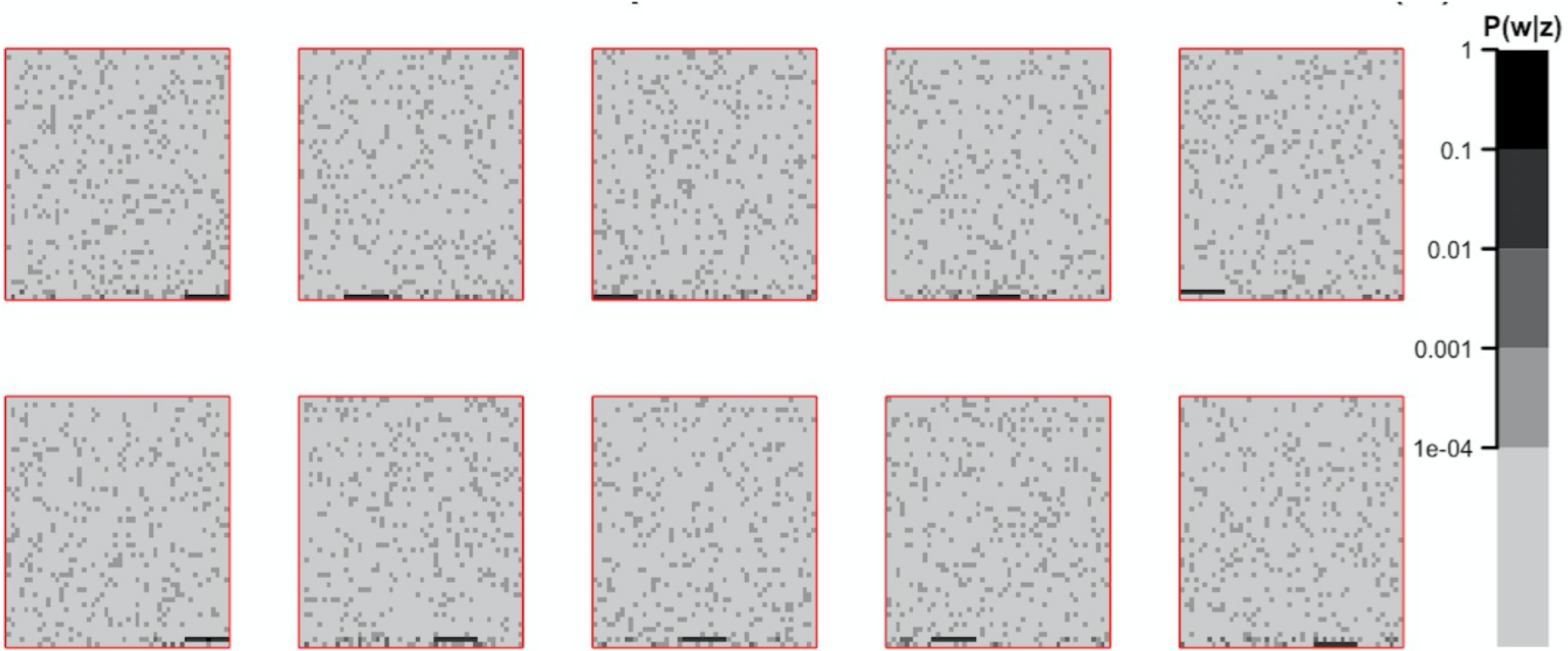
(Larger values are better)



Recall Topics:

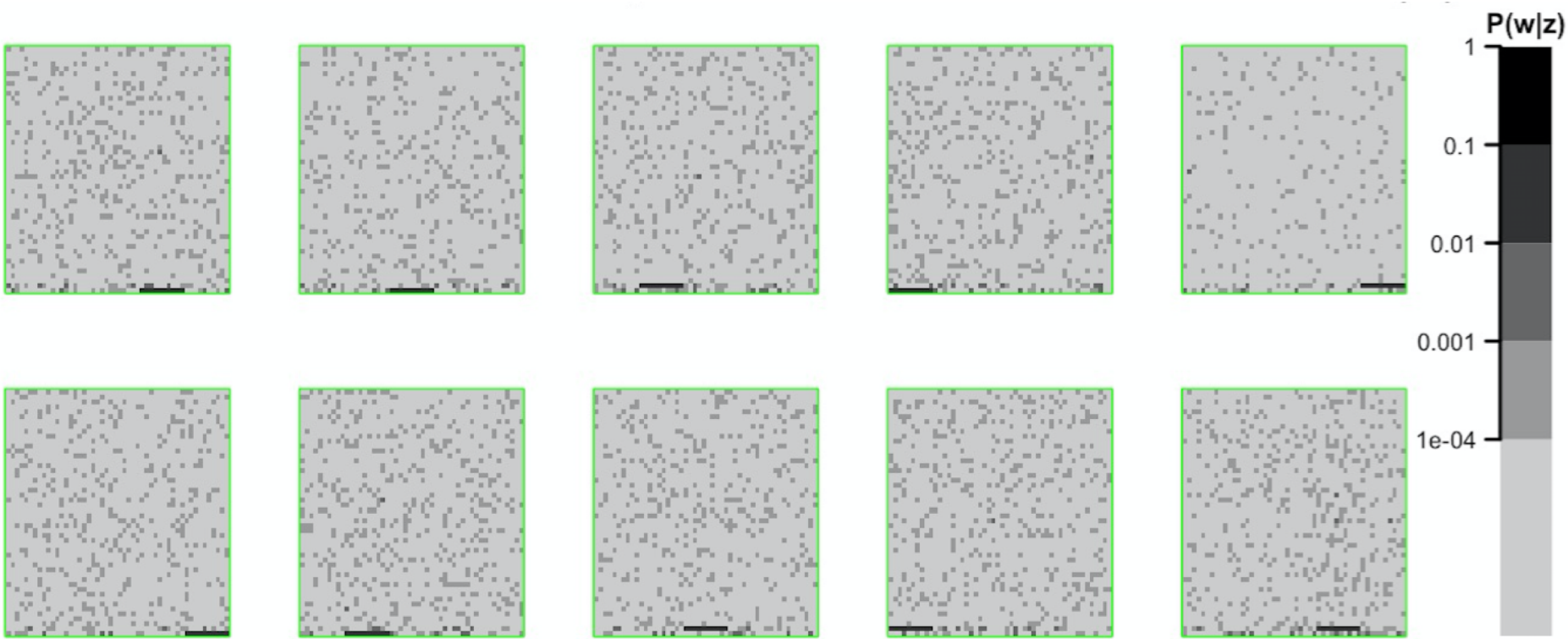


Results: Recovered Topics - without sketch



Results: Recovered Topics - with sketch

(memory use ⅘ of vocabulary: H = 2, J = 1000)



Conclusions and Future Directions

- Demonstrated incorporation of count-min sketch into inference algorithms for two ubiquitous probabilistic models.
- Useful when number of unique tokens (DM case) or vocabulary size (LDA) is too large to be stored in memory
- Were able to recover posterior probabilities in both cases:
  - DM-case: stream of M tokens. Were able to recover probability, θ_k, that k^{th} unique token appears in stream
  - LDA: were able to recover latent topics in toy corpus
- In future:
  - Applying LDA sketch-based inference on different document corpuses: corpuses with denser topics and on real-world corpuses
  - Investigating additional sketches: we are interested in proportions, not counts. Are there sketches that are better at preserving proportions?
  - Applying LDA sketch-based inference in streaming setting

Thanks!

Dirichlet-multinomial (DM) model

Multinomial occupancy scheme: K bins, M balls thrown independently in each bin
Probability θ_k of landing into the k^{th} bin

Let $x_1,...,x_M$ be M tokens in [K]. Likelihood has a **Categorical(θ)** distribution:

$$p(x_1,...,x_M \mid \theta) = \prod_n \prod_k \theta_k^{1(x_m=k)} = \prod_k \theta_k^{c_k}, \text{ where } c_k := \text{\# of balls in } k^{th} \text{ bin}$$

Under the DM model, we assume that the probabilities are random: $\theta \sim$ **Dirichlet$_K$(α)**

$\theta \in \Delta^K$

Goal is to compute the posterior distribution using Bayes' rule:

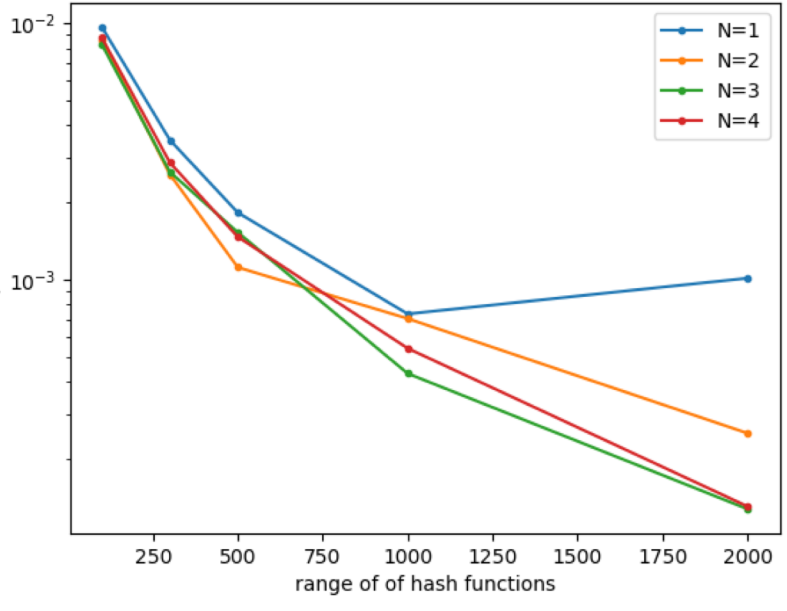
$$p(\theta \mid x_1,...,x_M) = p(\theta)\, p(x_1,...,x_M \mid \theta) = \text{Dirichlet}(\theta \mid \alpha + c), \quad \text{where } c = (c_1,...,c_k)$$

Predictive distribution: $p(x_{M+1} = k \mid x_1,...,x_M) = \int p(x_{M+1})\, p(\theta \mid x_1,...,x_M)\, d\theta$

CM sketch simulations on multinomial tokens

M=10,000; K=10,000

N hash functions, J range



For 1 trial: