Probabilistic Models in MapReduce

Jordan Boyd-Graber

April 7, 2011

UNIVERSITY OF MARYLAND 1856
COLLEGE OF INFORMATION STUDIES

Adapted from Jimmy Lin's Slides

Roadmap

- Homework
- Midterm
- Probabilistic Models
- Hidden Markov Model
- Topic Models

Homework

- Project Proposal: Due 11
- Homework 3?
- Homework 4 is out
- Homework 6

Midterm: Max

- Obvious answer
- Less obvious answer
  - Define grouper
  - Make sure values arrive in sorted order
  - Reducer only needs to look at first value
  - Not as efficient as using combiners (except in pathological situations)

Rule-Based Systems

Until the 1990s, text processing relied on rule-based systems

Advantages

- More predictable
- Easy to understand
- Easy to identify errors and fix them

Disadvantages

- Extremely labor-intensive to create
- Not robust to out of domain input
- No partial output or analysis when failure occurs

Statistical Methods

Basic idea: learn from a large corpus of examples of what we wish to model **Training Data**

Advantages

- More robust to the complexities of real-world input
- Creating training data is usually cheaper than creating rules
- Even easier today thanks to Amazon Mechanical Turk
- Data may already exist for independent reasons

Disadvantages

- Systems often behave differently compared to expectations
- Hard to understand the reasons for errors or debug errors

- Learning from training data usually means estimating the parameters of the statistical model
- Estimation usually carried out via machine learning
- Two kinds of machine learning algorithms
  - **Supervised learning**
    - Training data consists of the inputs and respective outputs (labels)
    - Labels are usually created via expert annotation (expensive)
    - Difficult to annotate when predicting more complex outputs
  - **Unsupervised learning**
    - Training data just consists of inputs. No labels.
    - One example of such an algorithm: Expectation Maximization (EM)

What Problems Can We Solve?

- (Supervised) Part of speech tagging
- (Unsupervised) Exploring large corpora

What Problems Can We Solve?

- (Supervised) Part of speech tagging
- (Unsupervised) Exploring large corpora
- But first, a brief recap of estimating probability distributions

How do we estimate a probability?

- Suppose we want to estimate $P(w_n = \text{"dog"} | z_z = \text{"NN"})$.

How do we estimate a probability?

- Suppose we want to estimate $P(w_n = \text{“dog”} | z_z = \text{“NN”})$.

dog	**dog**	cat	horse	cow
cat	horse	cow	fly	mouse
fly	**dog**	cat	fly	**dog**
mouse	**dog**	fly	cat	cow

How do we estimate a probability?

- Suppose we want to estimate $P(w_n = \text{"dog"} | z_z = \text{"NN"})$.

dog	**dog**	cat	horse	cow
cat	horse	cow	fly	mouse
fly	**dog**	cat	fly	**dog**
mouse	**dog**	fly	cat	cow

- Maximum likelihood (ML) estimate of the probability is:

$$\hat{\theta}_i = \frac{n_i}{\sum_k n_k} \tag{1}$$

How do we estimate a probability?

- Suppose we want to estimate $P(w_n = \text{“dog”} | z_z = \text{“NN”})$.

dog	**dog**	cat	horse	cow
cat	horse	cow	fly	mouse
fly	**dog**	cat	fly	**dog**
mouse	**dog**	fly	cat	cow

- Maximum likelihood (ML) estimate of the probability is:

$$\hat{\theta}_i = \frac{n_i}{\sum_k n_k} \tag{1}$$

- Is this reasonable?

How do we estimate a probability?

- In computational linguistics, we often have a *prior* notion of what our probability distributions are going to look like (for example, non-zero, sparse, uniform, etc.).
- This estimate of a probability distribution is called the maximum a posteriori (MAP) estimate:

$$\theta_{\text{MAP}} = \text{argmax}_{\theta} f(x|\theta) g(\theta) \tag{2}$$

How do we estimate a probability?

- For a multinomial distribution (i.e. a discrete distribution, like over words):

$$\theta_i = \frac{n_i + \alpha_i}{\sum_k n_k + \alpha_k} \tag{3}$$

- α_i is called a smoothing factor, a pseudocount, etc.

How do we estimate a probability?

- For a multinomial distribution (i.e. a discrete distribution, like over words):
$$\theta_i = \frac{n_i + \alpha_i}{\sum_k n_k + \alpha_k} \tag{3}$$
- α_i is called a smoothing factor, a pseudocount, etc.
- When $\alpha_i = 1$ for all i, it's called "Laplace smoothing" and corresponds to a uniform prior over all multinomial distributions (we talked about this before).

How do we estimate a probability?

- For a multinomial distribution (i.e. a discrete distribution, like over words):

$$\theta_i = \frac{n_i + \alpha_i}{\sum_k n_k + \alpha_k} \tag{3}$$

- α_i is called a smoothing factor, a pseudocount, etc.
- When $\alpha_i = 1$ for all i, it's called "Laplace smoothing" and corresponds to a uniform prior over all multinomial distributions (we talked about this before).
- To geek out, the set $\{\alpha_1, \ldots, \alpha_N\}$ parameterizes a Dirichlet distribution, which is itself a distribution over distributions and is the conjugate prior of the Multinomial (more later).

Parts of Speech

- The Art of Grammar circa 100 B.C.
- Written to allow post-Classical Greek speakers to understand Odyssey and other classical poets

[Noun, Verb, Pronoun, Article, Adverb, Conjunction, Participle, Preposition]

- Remarkably enduring list
- Occur in almost every language
- Defined primarily in terms of syntactic and morphological criteria (affixes)

Categories of POS Tags

Closed Class

- Relatively fixed membership
- Conjunctions, Prepositions, Auxiliaries, Determiners, Pronouns
- Function words: short and used primarily for structuring

Open Class

- Nouns, Verbs, Adjectives, Adverbs
- Frequent neologisms (borrowed/coined)
- Most types

Tagsets

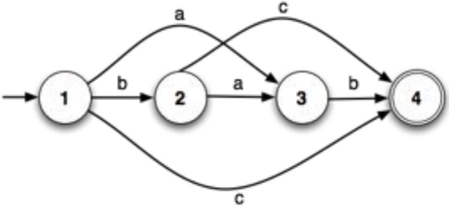
- Several English tagsets have been developed (language specific)
- Vary in number of tags
- Brown Tagset (87)
- Penn Treebank (45) [More common]
- Simple morphology = more ambiguity = smaller tagset
- Size depends on language and purpose

Why?

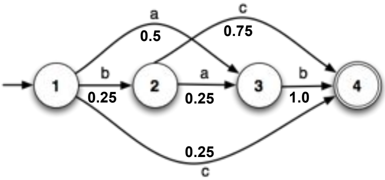
- Corpus-based Linguistic Analysis & Lexicography
- Information Retrieval & Question Answering
- Automatic Speech Synthesis
- Word Sense Disambiguation
- Shallow Syntactic Parsing
- Machine Translation

What do we need to specify an FSM formally?

- Finite number of states
- Transitions
- Input alphabet
- Start state
- Final state(s)



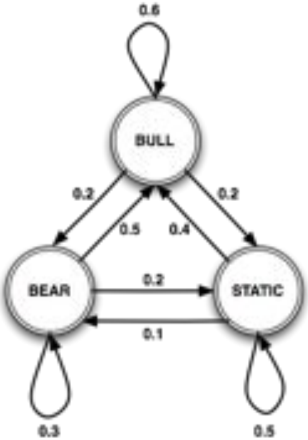
Weighted FSM



- a' is twice as likely to be seen in state 1 as b' or c'
- c' is three times as likely to be seen in state 2 as a'

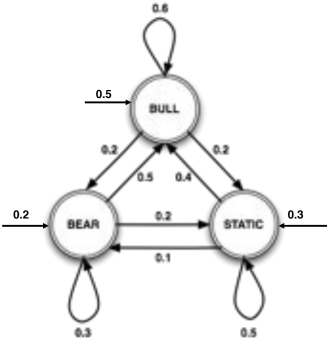
$$P(ab') = 0.50 * 1.00 = 0.5, P(bc') = 0.25 * 0.75 = 0.1875 \quad (4)$$

Observable States and Probabilistic Emissions



- This not a valid prob. FSM!
- No start states

Observable States and Probabilistic Emissions



- This not a valid prob. FSM!
- No start states
- Use prior probabilities
- Note that prob. of being in any state ONLY depends on previous state ,i.e., the (1^{st} order) Markov assumption
- This extension of a prob. FSM is called a Markov Chain or an Observed Markov Model
- Each state corresponds to an observable physical event

Are states observable?

Day: 1, 2, 3, 4, 5, 6

$\uparrow \quad \downarrow \quad \leftrightarrow \quad \uparrow \quad \downarrow \quad \leftrightarrow$

$\uparrow$: **Market is up**
$\downarrow$: **Market is down**
$\leftrightarrow$: **Market hasn't changed**

What you actually observe:

Are states observable?

Day: 1, 2, 3, 4, 5, 6

$\uparrow \; \downarrow \; \leftrightarrow \; \uparrow \; \downarrow \; \leftrightarrow$

$\uparrow$: Market is up
$\downarrow$: Market is down
$\leftrightarrow$: Market hasn't changed

What you actually observe:

Day: 1, 2, 3, 4, 5, 6

Bu, Be, S, Be, S, Bu

Bu: Bull Market
Be: Bear Market
S : Static Market

HMM Intuitions

- Need to model problems where observed events don't correspond to states directly
- Instead observations are probabilistic of *hidden* state
- Solution: A Hidden Markov Model (HMM)
- Assume two probabilistic processes
  - Underlying process is hidden (states = hidden events)
  - Second process produces sequence of observed events

HMM Definition

Assume K parts of speech, a lexicon size of V, a series of observations $\{x_1, \dots, x_N\}$, and a series of unobserved states $\{z_1, \dots, z_N\}$.

π A distribution over start states (vector of length K): $\pi_i = p(z_1 = i)$

θ Transition matrix (matrix of size K by K): $\beta_{i,j} = p(z_n = j | z_{n-1} = i)$

β An emission matrix (matrix of size K by V): $\beta_{k,v} = p(x_n = v | z_n = k)$

HMM Definition

Assume K parts of speech, a lexicon size of V, a series of observations $\{x_1, \dots, x_N\}$, and a series of unobserved states $\{z_1, \dots, z_N\}$.

π A distribution over start states (vector of length K): $\pi_i = p(z_1 = i)$

θ Transition matrix (matrix of size K by K): $\beta_{i,j} = p(z_n = j | z_{n-1} = i)$

β An emission matrix (matrix of size K by V): $\beta_{k,v} = p(x_n = v | z_n = k)$

Two problems: How do we move from data to a model? (Estimation) How do we move from a model and unlabled data to labeled data? (Inference)

Training Sentences

here	come	old	flattop
MOD	V	MOD	N

a	crowd	of	people	stopped	and	stared
DET	N	PREP	N	V	CONJ	V

gotta	get	you	into	my	life
V	V	PRO	PREP	PRO	V

and	I	love	her
CONJ	PRO	V	PRO

Initial Probability π

POS	Frequency	Probability
MOD	1.1	0.234
DET	1.1	0.234
CONJ	1.1	0.234
N	0.1	0.021
PREP	0.1	0.021
PRO	0.1	0.021
V	1.1	0.234

Remember, we're taking MAP estimates, so we add 0.1 (arbitrarily chosen) to each of the counts before normalizing to create a probability distribution. This is easy; one sentence starts with an adjective, one with a determiner, one with a verb, and one with a conjunction.

Transition Probability θ

- We can ignore the words; just look at the parts of speech. Let's compute one row, the row for verbs.
- We see the following transitions: V $\rightarrow$ MOD, V $\rightarrow$ CONJ, V $\rightarrow$ V, V $\rightarrow$ PRO, and V $\rightarrow$ PRO

POS	Frequency	Probability
MOD	1.1	0.193
DET	0.1	0.018
CONJ	1.1	0.193
N	0.1	0.018
PREP	0.1	0.018
PRO	2.1	0.368
V	1.1	0.193

- And do the same for each part of speech ...

Emission Probability β

Let's look at verbs . . .

Word	a	and	come	crowd	flattop
Frequency	0.1	0.1	1.1	0.1	0.1
Probability	0.011	0.011	0.121	0.011	0.011

Word	get	gotta	her	here	i
Frequency	1.1	1.1	0.1	0.1	0.1
Probability	0.121	0.121	0.011	0.011	0.011

Word	into	it	life	love	my
Frequency	0.1	0.1	0.1	1.1	0.1
Probability	0.011	0.011	0.011	0.121	0.011

Word	of	old	people	stared	stood
Frequency	0.1	0.1	0.1	1.1	1.1
Probability	0.011	0.011	0.011	0.121	0.121

Viterbi Algorithm

- Given an unobserved sequence of length L, $\{x_1, \ldots, x_L\}$, we want to find a sequence $\{z_1 \ldots z_L\}$ with the highest probability.

Viterbi Algorithm

- Given an unobserved sequence of length L, $\{x_1, \ldots, x_L\}$, we want to find a sequence $\{z_1 \ldots z_L\}$ with the highest probability.
- It's impossible to compute K^L possibilities.
- So, we use dynamic programming to compute best sequence for each subsequence from 0 to l.
- Base case:

$$\delta_1(k) = \pi_k \beta_{k,x_i} \tag{5}$$

- Recursion:

$$\delta_n(k) = \max_j (\delta_{n-1}(j)\theta_{j,k})\beta_{k,x_n} \tag{6}$$

- The complexity of this is now K^2L.
- But just computing the max isn't enough. We also have to remember where we came from. (Breadcrumbs from best previous state.)

$$\Psi_n = \text{argmax}_j \delta_{n-1}(j)\theta_{j,k} \tag{7}$$

- The complexity of this is now $K^2 L$.
- But just computing the max isn't enough. We also have to remember where we came from. (Breadcrumbs from best previous state.)

$$\Psi_n = \text{argmax}_j \delta_{n-1}(j)\theta_{j,k} \tag{7}$$

- Let's do that for the sentence "come and get it"

POS	π_k	β_{k,x_1}	$\log \delta_1(k)$
MOD	0.234	0.024	-5.18
DET	0.234	0.032	-4.89
CONJ	0.234	0.024	-5.18
N	0.021	0.016	-7.99
PREP	0.021	0.024	-7.59
PRO	0.021	0.016	-7.99
V	0.234	0.121	-3.56

come and get it

Why logarithms?

1. More interpretable than a float with lots of zeros.
2. Underflow is less of an issue
3. Addition is cheaper than multiplication

POS	$\log \delta_1(j)$		$\log \delta_1(\text{CONJ})$
MOD	-5.18		
DET	-4.89		
CONJ	-5.18		
N	-7.99		
PREP	-7.59		
PRO	-7.99		
V	-3.56		

come **and** get it

POS	$\log \delta_1(j)$		$\log \delta_1(\text{CONJ})$
MOD	-5.18		
DET	-4.89		
CONJ	-5.18		???
N	-7.99		
PREP	-7.59		
PRO	-7.99		
V	-3.56		

come **and** get it

POS	$\log \delta_1(j)$	$\log \delta_1(j)\theta_{j,\text{CONJ}}$	$\log \delta_1(\text{CONJ})$
MOD	-5.18		
DET	-4.89		
CONJ	-5.18		???
N	-7.99		
PREP	-7.59		
PRO	-7.99		
V	-3.56		

come **and** get it

POS	$\log \delta_1(j)$	$\log \delta_1(j)\theta_{j,\text{CONJ}}$	$\log \delta_1(\text{CONJ})$
MOD	-5.18		
DET	-4.89		
CONJ	-5.18		???
N	-7.99		
PREP	-7.59		
PRO	-7.99		
V	-3.56		

come **and** get it

$$\log\left(\delta_0(\text{V})\theta_{\text{V, CONJ}}\right) = \log \delta_0(k) + \log \theta_{\text{V, CONJ}} = -3.56 + -1.65$$

POS	$\log \delta_1(j)$	$\log \delta_1(j)\theta_{j,\text{CONJ}}$	$\log \delta_1(\text{CONJ})$
MOD	-5.18		
DET	-4.89		
CONJ	-5.18		???
N	-7.99		
PREP	-7.59		
PRO	-7.99		
V	-3.56	-5.21	

come **and** get it

POS	$\log \delta_1(j)$	$\log \delta_1(j)\theta_{j,\text{CONJ}}$	$\log \delta_1(\text{CONJ})$
MOD	-5.18		
DET	-4.89		
CONJ	-5.18		???
N	-7.99	≤ -7.99	
PREP	-7.59	≤ -7.59	
PRO	-7.99	≤ -7.99	
V	-3.56	-5.21	

come **and** get it

POS	$\log \delta_1(j)$	$\log \delta_1(j)\theta_{j,\text{CONJ}}$	$\log \delta_1(\text{CONJ})$
MOD	-5.18	-8.48	
DET	-4.89	-7.72	
CONJ	-5.18	-8.47	???
N	-7.99	≤ -7.99	
PREP	-7.59	≤ -7.59	
PRO	-7.99	≤ -7.99	
V	-3.56	-5.21	

come **and** get it

POS	$\log \delta_1(j)$	$\log \delta_1(j)\theta_{j,\text{CONJ}}$	$\log \delta_1(\text{CONJ})$
MOD	-5.18	-8.48	
DET	-4.89	-7.72	
CONJ	-5.18	-8.47	???
N	-7.99	≤ -7.99	
PREP	-7.59	≤ -7.59	
PRO	-7.99	≤ -7.99	
V	-3.56	-5.21	

come **and** get it

POS	$\log \delta_1(j)$	$\log \delta_1(j)\theta_{j,\text{CONJ}}$	$\log \delta_1(\text{CONJ})$
MOD	-5.18	-8.48	
DET	-4.89	-7.72	
CONJ	-5.18	-8.47	
N	-7.99	≤ -7.99	
PREP	-7.59	≤ -7.59	
PRO	-7.99	≤ -7.99	
V	-3.56	-5.21	

come **and** get it

$\log \delta_1(k) = -5.21 + \log \beta_{\text{CONJ, and}} =$

POS	$\log \delta_1(j)$	$\log \delta_1(j)\theta_{j,\text{CONJ}}$	$\log \delta_1(\text{CONJ})$
MOD	-5.18	-8.48	
DET	-4.89	-7.72	
CONJ	-5.18	-8.47	
N	-7.99	≤ -7.99	
PREP	-7.59	≤ -7.59	
PRO	-7.99	≤ -7.99	
V	-3.56	-5.21	

come **and** get it

$$\log \delta_1(k) = -5.21 + \log \beta_{\text{CONJ, and}} = \; -5.21 - 0.81$$

POS	$\log \delta_1(j)$	$\log \delta_1(j)\theta_{j,\text{CONJ}}$	$\log \delta_1(\text{CONJ})$
MOD	-5.18	-8.48	
DET	-4.89	-7.72	
CONJ	-5.18	-8.47	-6.02
N	-7.99	≤ -7.99	
PREP	-7.59	≤ -7.59	
PRO	-7.99	≤ -7.99	
V	-3.56	-5.21	

come **and** get it

POS	$\delta_1(k)$	$\delta_2(k)$	b_2	$\delta_3(k)$	b_3	$\delta_4(k)$	b_4
MOD	-5.18						
DET	-4.89						
CONJ	-5.18	-6.02	V				
N	-7.99						
PREP	-7.59						
PRO	-7.99						
V	-3.56						
WORD	come	and		get		it	

POS	$\delta_1(k)$	$\delta_2(k)$	b_2	$\delta_3(k)$	b_3	$\delta_4(k)$	b_4
MOD	-5.18	-0.00	X				
DET	-4.89	-0.00	X				
CONJ	-5.18	-6.02	V				
N	-7.99	-0.00	X				
PREP	-7.59	-0.00	X				
PRO	-7.99	-0.00	X				
V	-3.56	-0.00	X				
WORD	come	and		get		it	

POS	$\delta_1(k)$	$\delta_2(k)$	b_2	$\delta_3(k)$	b_3	$\delta_4(k)$	b_4
MOD	-5.18	-0.00	X	-0.00	X		
DET	-4.89	-0.00	X	-0.00	X		
CONJ	-5.18	-6.02	V	-0.00	X		
N	-7.99	-0.00	X	-0.00	X		
PREP	-7.59	-0.00	X	-0.00	X		
PRO	-7.99	-0.00	X	-0.00	X		
V	-3.56	-0.00	X	-9.03	CONJ		
WORD	come	and		get		it	

POS	$\delta_1(k)$	$\delta_2(k)$	b_2	$\delta_3(k)$	b_3	$\delta_4(k)$	b_4
MOD	-5.18	-0.00	X	-0.00	X	-0.00	X
DET	-4.89	-0.00	X	-0.00	X	-0.00	X
CONJ	-5.18	-6.02	V	-0.00	X	-0.00	X
N	-7.99	-0.00	X	-0.00	X	-0.00	X
PREP	-7.59	-0.00	X	-0.00	X	-0.00	X
PRO	-7.99	-0.00	X	-0.00	X	-14.6	V
V	-3.56	-0.00	X	-9.03	CONJ	-0.00	X
WORD	come	and		get		it	

MapReduce: HMM Learning

Mapper

```
def map(sentence_id, sentence):
  prev = None
  for state, word in sentence:
    if prev == None:
      emit(("S", 0, -1), 1)
      emit(("S", 0, state), 1)
    else:
       emit(("T", prev, word), 1)
       emit(("T", prev, word), 1)
     emit(("E", state, word), 1)
     emit(("E", state, word), 1)
```

MapReduce: HMM Learning

Reducer

```
def reduce(key, values):
  distribution, i, j = key
  if j == -1:
     normalizer = sum(values) +
                           priors_sum[distribution]
  else:
     emit key, (sum(values) +
                 priors_element[distribution]) /
                 normalizer
```

MapReduce: HMM Testing

- Distributed parameters via distributed cache
- Do Vitterbi for each sentence using parameters
- Can happen independently in a mapper, output final assignment
- Use identity reducer
- Output final POS sequence (See Lin & Dyer for the gory details)

Why topic models?

- Suppose you have a huge number of documents
- You want to know what's going on
- Don't have time to read them (e.g. every New York Times article from the 50's)
- Topic models offer a way to get a corpus-level view of major themes

Why topic models?

- Suppose you have a huge number of documents
- You want to know what's going on
- Don't have time to read them (e.g. every New York Times article from the 50's)
- Topic models offer a way to get a corpus-level view of major themes
- Unsupervised

Conceptual Approach

- Given a corpus, what topics (a priori number) are expressed throughout the corpus?

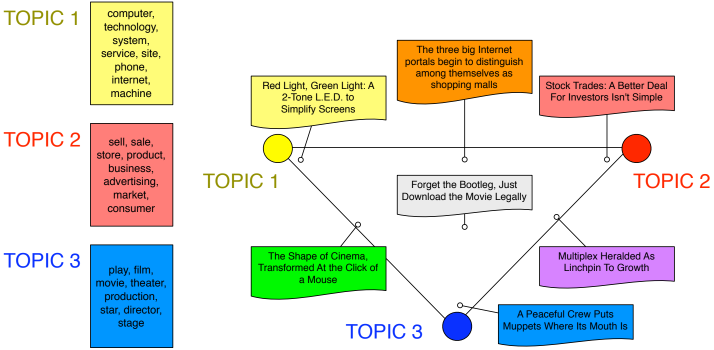
Conceptual Approach

- Given a corpus, what topics (a priori number) are expressed throughout the corpus?

TOPIC 1	computer, technology, system, service, site, phone, internet, machine
TOPIC 2	sell, sale, store, product, business, advertising, market, consumer
TOPIC 3	play, film, movie, theater, production, star, director, stage

Conceptual Approach

- Given a corpus, what topics (a priori number) are expressed throughout the corpus?
- For each document, what topics are expressed by that document?



Topics from *Science*

human	evolution	disease	computer
genome	evolutionary	host	models
dna	species	bacteria	information
genetic	organisms	diseases	data
genes	life	resistance	computers
sequence	origin	bacterial	system
gene	biology	new	network
molecular	groups	strains	systems
sequencing	phylogenetic	control	model
map	living	infectious	parallel
information	diversity	malaria	methods
genetics	group	parasite	networks
mapping	new	parasites	software
project	two	united	new
sequences	common	tuberculosis	simulations

Why should you care?

- Neat way to explore / understand corpus collections
- NLP Applications
  - POS Tagging [*Toutanova and Johnson* 2008]
  - Word Sense Disambiguation [*Boyd-Graber et al.* 2007]
  - Word Sense Induction [*Brody and Lapata* 2009]
  - Discourse Segmentation [*Purver et al.* 2006]
- Psychology [*Griffiths et al.* 2007b]: word meaning, polysemy
- Inference is (relatively) simple

Matrix Factorization Approach

$$\underbrace{\begin{bmatrix} M \times K \end{bmatrix}}_{\text{Topic Assignment}} \times \underbrace{\begin{bmatrix} K \times V \end{bmatrix}}_{\text{Topics}} \approx \underbrace{\begin{bmatrix} M \times V \end{bmatrix}}_{\text{Dataset}}$$

- K Number of topics
- M Number of documents
- V Size of vocabulary

Matrix Factorization Approach

$$\underset{\text{Topic Assignment}}{\begin{bmatrix} \\ M \times K \\ \\ \end{bmatrix}} \times \underset{\text{Topics}}{\begin{bmatrix} \quad K \times V \quad \end{bmatrix}} \approx \underset{\text{Dataset}}{\begin{bmatrix} \\ \quad M \times V \quad \\ \\ \end{bmatrix}}$$

- **K** Number of topics
- **M** Number of documents
- **V** Size of vocabulary

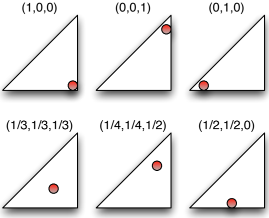
- If you use singular value decomposition (SVD), this technique is called latent semantic analysis.
- Popular in information retrieval.

Alternative: Generative Model

- How your data came to be
- Sequence of Probabilistic Steps
- Posterior Inference

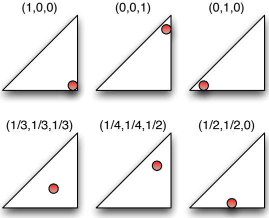
Multinomial Distribution

- Distribution over discrete outcomes
- Represented by non-negative vector that sums to one
- Picture representation



Multinomial Distribution

- Distribution over discrete outcomes
- Represented by non-negative vector that sums to one
- Picture representation



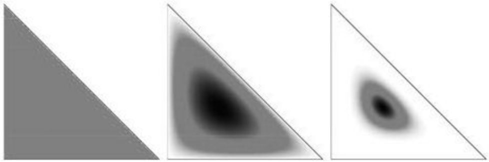
- Come from a Dirichlet distribution

Dirichlet Distribution

$$P(\boldsymbol{p} \mid \alpha \boldsymbol{m}) = \frac{\Gamma(\sum_k \alpha m_k)}{\prod_k \Gamma(\alpha m_k)} \prod_k p_k^{\alpha m_k - 1}$$

Dirichlet Distribution

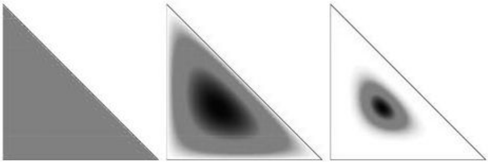
$$P(\boldsymbol{p} \mid \alpha\boldsymbol{m}) = \frac{\Gamma(\sum_k \alpha m_k)}{\prod_k \Gamma(\alpha m_k)} \prod_k p_k^{\alpha m_k - 1}$$



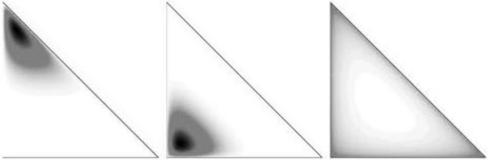
$\alpha = 3, \boldsymbol{m} = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$ $\alpha = 6, \boldsymbol{m} = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$ $\alpha = 30, \boldsymbol{m} = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$

Dirichlet Distribution

$$P(\boldsymbol{p} \mid \alpha \boldsymbol{m}) = \frac{\Gamma(\sum_k \alpha m_k)}{\prod_k \Gamma(\alpha m_k)} \prod_k p_k^{\alpha m_k - 1}$$

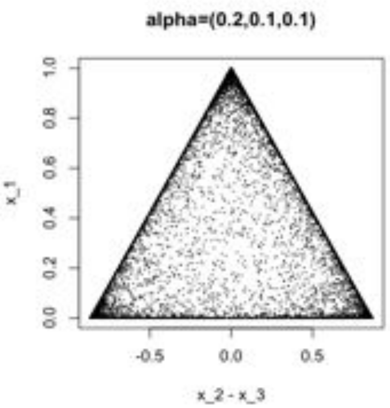


$\alpha = 3, \boldsymbol{m} = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$ $\alpha = 6, \boldsymbol{m} = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$ $\alpha = 30, \boldsymbol{m} = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$

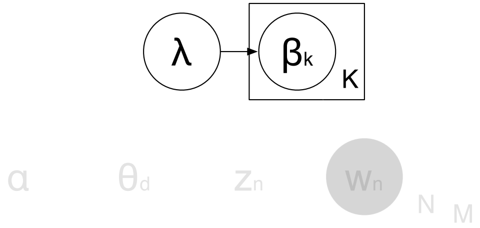


$\alpha = 14, \boldsymbol{m} = (\frac{1}{7}, \frac{5}{7}, \frac{1}{7})$ $\alpha = 14, \boldsymbol{m} = (\frac{1}{7}, \frac{1}{7}, \frac{5}{7})$ $\alpha = 2.7, \boldsymbol{m} = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$

Dirichlet Distribution

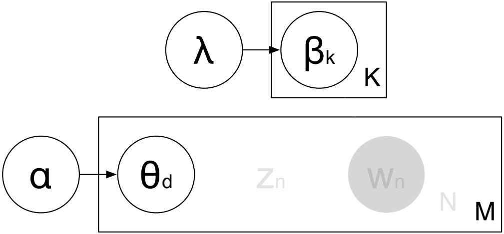


Generative Model Approach



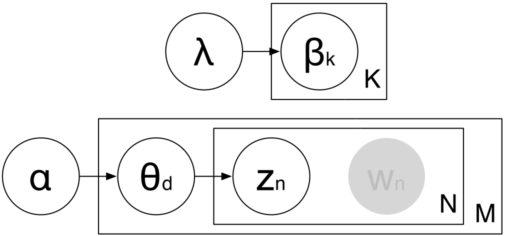
- For each topic $k \in \{1, \dots, K\}$, draw a multinomial distribution β_k from a Dirichlet distribution with parameter λ

Generative Model Approach



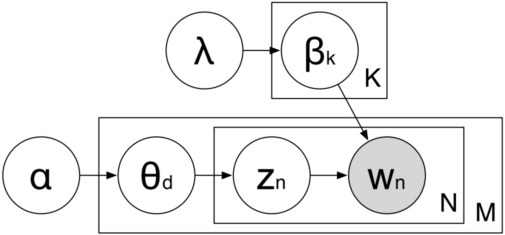
- For each topic $k \in \{1, \ldots, K\}$, draw a multinomial distribution β_k from a Dirichlet distribution with parameter λ
- For each document $d \in \{1, \ldots, M\}$, draw a multinomial distribution θ_d from a Dirichlet distribution with parameter α

Generative Model Approach



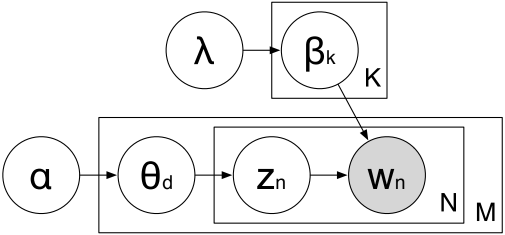
- For each topic $k \in \{1, \ldots, K\}$, draw a multinomial distribution β_k from a Dirichlet distribution with parameter λ
- For each document $d \in \{1, \ldots, M\}$, draw a multinomial distribution θ_d from a Dirichlet distribution with parameter α
- For each word position $n \in \{1, \ldots, N\}$, select a hidden topic z_n from the multinomial distribution parameterized by θ.

Generative Model Approach



- For each topic $k \in \{1, \dots, K\}$, draw a multinomial distribution β_k from a Dirichlet distribution with parameter λ
- For each document $d \in \{1, \dots, M\}$, draw a multinomial distribution θ_d from a Dirichlet distribution with parameter α
- For each word position $n \in \{1, \dots, N\}$, select a hidden topic z_n from the multinomial distribution parameterized by θ.
- Choose the observed word w_n from the distribution β_{z_n}.

Generative Model Approach



- For each topic $k \in \{1, \ldots, K\}$, draw a multinomial distribution β_k from a Dirichlet distribution with parameter λ
- For each document $d \in \{1, \ldots, M\}$, draw a multinomial distribution θ_d from a Dirichlet distribution with parameter α
- For each word position $n \in \{1, \ldots, N\}$, select a hidden topic z_n from the multinomial distribution parameterized by θ.
- Choose the observed word w_n from the distribution β_{z_n}.

We use statistical inference to uncover the most likely unobserved

Topic Models: What's Important

- A generative probabilistic model of document collections that posits a hidden topical structure which is inferred from data
- A topic is a distribution over words
- Have semantic coherence because of language use
- We use latent Dirichlet allocation (LDA) [*Blei et al.* 2003], a fully Bayesian version of pLSI [*Hofmann* 1999]

Learning topics

- What we want: a (topic) model
- This is represented by a configuration latent variables z
- What we have: our data D, any hyperparameters Ξ
- Compute likelihood $L = p(D|z, \Xi)$.
- Higher this number is, the better we're doing

Expectation Maximization Algorithm

- Input: z (hidden variables), ξ (parameters), D (data)
- Start with initial guess of z
- Repeat
  - Compute the parameters ξ that maximize likelihood L (use calculus)
  - Compute the expected value of latent variables z
- With each iteration, objective function goes up

Expectation Maximization Algorithm

- Input: z (hidden variables), ξ (parameters), D (data)
- Start with initial guess of z
- Repeat
  - Compute the parameters ξ that maximize likelihood L (use calculus)
  - **E-Step** Compute the expected value of latent variables z
- With each iteration, objective function goes up

Expectation Maximization Algorithm

- Input: z (hidden variables), ξ (parameters), D (data)
- Start with initial guess of z
- Repeat
  - **M-Step** Compute the parameters ξ that maximize likelihood L (use calculus)
  - **E-Step** Compute the expected value of latent variables z
- With each iteration, objective function goes up

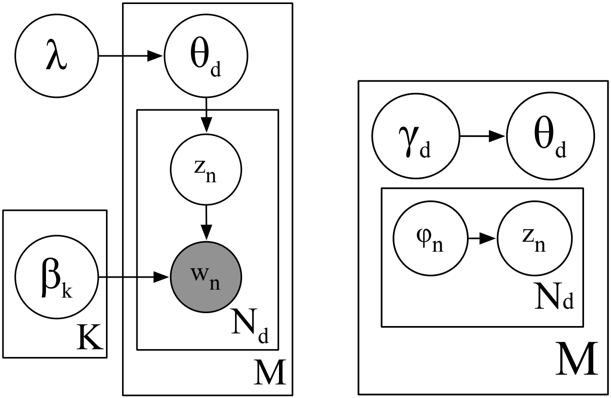
Theory

- Sometimes you can't actually optimize L
- So we instead optimize a lower bound based on a "variational" distribution q

$$\mathcal{L} = \mathbb{E}_q \left[\log \left(p(\mathrm{D}|Z)p(Z|\xi)\right)\right] - \mathbb{E}_q \left[\log q(Z)\right] \tag{8}$$

- $L - \mathcal{L} = KL(p||q)$
- This is called variational EM (normal EM is when $p = q$)
- Makes the math possible to optimize $\mathcal{L}$

Variational distribution



(a) LDA

(b) Variational

Updates - Important Part

- ϕ How much the n^{th} word in a document expressed topic k
- $\gamma_{d,k}$ How much the k^{th} topic is expressed in a document d
- $\beta_{v,k}$ How much word v is associated with topic k

$$\phi_{d,n,k} \propto \beta_{w_{d,n},k} \cdot e^{\Psi(\gamma_k)}$$

$$\gamma_{d,k} = \alpha_k + \sum_{n=1}^{N_d} \phi_{d,n,k},$$

$$\beta_{v,k} \propto \sum_{d=1}^{C} (w_v^{(d)} \phi_{d,v,k})$$

This is the algorithm!

Updates - Important Part

- ϕ How much the n^{th} word in a document expressed topic k
- $\gamma_{d,k}$ How much the k^{th} topic is expressed in a document d
- $\beta_{v,k}$ How much word v is associated with topic k

$$\phi_{d,n,k} \propto \beta_{w_{d,n},k} \cdot e^{\Psi(\gamma_k)}$$

$$\gamma_{d,k} = \alpha_k + \sum_{n=1}^{N_d} \phi_{d,n,k},$$

$$\beta_{v,k} \propto \sum_{d=1}^{C} (w_v^{(d)} \phi_{d,v,k})$$

This is the algorithm!

Objective Function

Expanding Equation 8 gives us $\mathcal{L}(\gamma, \phi; \alpha, \beta)$ for one document:

$$
\begin{aligned}
\mathcal{L}(\gamma, \phi; \alpha, \beta) &= \sum_{d=1}^{C} \mathcal{L}_d(\gamma, \phi; \alpha, \beta) \\
&= \underbrace{\sum_{d=1}^{C} \mathcal{L}_d(\alpha)}_{\text{Driver}} + \underbrace{\sum_{d=1}^{C} (\underbrace{\mathcal{L}_d(\gamma, \phi) + \mathcal{L}_d(\phi) + \mathcal{L}_d(\gamma)}_{\text{computed in mapper}})}_{\text{computed in Reducer}},
\end{aligned}
$$

where

$$
\mathcal{L}_d(\alpha) = \log \Gamma\left(\textstyle\sum_{k=1}^{K} \alpha_k\right) - \sum_{i=1}^{K} \log \Gamma\left(\alpha_k\right),
$$

Objective Function

Expanding Equation 8 gives us $\mathcal{L}(\gamma, \phi; \alpha, \beta)$ for one document:

$$
\begin{aligned}
\mathcal{L}(\gamma, \phi; \alpha, \beta) &= \sum_{d=1}^{C} \mathcal{L}_d(\gamma, \phi; \alpha, \beta) \\
&= \underbrace{\sum_{d=1}^{C} \mathcal{L}_d(\alpha)}_{\text{Driver}} + \underbrace{\sum_{d=1}^{C} (\underbrace{\mathcal{L}_d(\gamma, \phi) + \mathcal{L}_d(\phi) + \mathcal{L}_d(\gamma)}_{\text{computed in mapper}})}_{\text{computed in Reducer}},
\end{aligned}
$$

where

$$
\mathcal{L}_d(\gamma, \phi) = \sum_{k=1}^{K} \left[\sum_{v=1}^{V} \phi_{v,k} - \sum_{v=1}^{V} \phi_{v,k} w_v \right] \left[\Psi(\gamma_k) - \Psi\left(\sum_{i=1}^{K} \gamma_i \right) \right],
$$

Objective Function

Expanding Equation 8 gives us $\mathcal{L}(\gamma, \phi; \alpha, \beta)$ for one document:

$$\begin{aligned}
\mathcal{L}(\gamma, \phi; \alpha, \beta) &= \sum_{d=1}^{C} \mathcal{L}_d(\gamma, \phi; \alpha, \beta) \\
&= \underbrace{\sum_{d=1}^{C} \mathcal{L}_d(\alpha)}_{\text{Driver}} + \underbrace{\sum_{d=1}^{C} (\underbrace{\mathcal{L}_d(\gamma, \phi) + \mathcal{L}_d(\phi) + \mathcal{L}_d(\gamma)}_{\text{computed in mapper}})}_{\text{computed in Reducer}},
\end{aligned}$$

where

$$\mathcal{L}_d(\phi) = \sum_{v=1}^{V} \sum_{k=1}^{K} \phi_{v,k} (\log \phi_{v,k} + \sum_{i=1}^{V} w_i \log \beta_{i,k}),$$

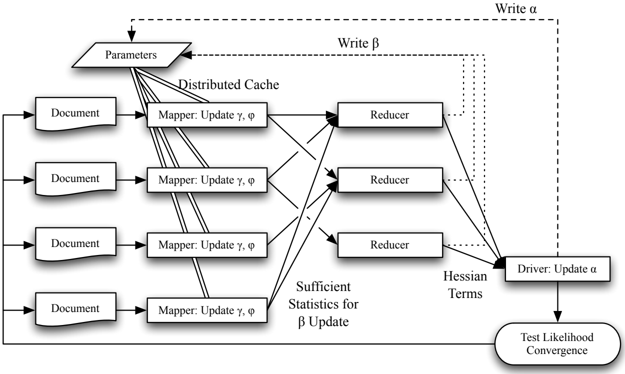
Objective Function

Expanding Equation 8 gives us $\mathcal{L}(\gamma, \phi; \alpha, \beta)$ for one document:

$$
\begin{aligned}
\mathcal{L}(\gamma, \phi; \alpha, \beta) &= \sum_{d=1}^{C} \mathcal{L}_d(\gamma, \phi; \alpha, \beta) \\
&= \underbrace{\sum_{d=1}^{C} \mathcal{L}_d(\alpha)}_{\text{Driver}} + \underbrace{\sum_{d=1}^{C} (\underbrace{\mathcal{L}_d(\gamma, \phi) + \mathcal{L}_d(\phi) + \mathcal{L}_d(\gamma)}_{\text{computed in mapper}})}_{\text{computed in Reducer}},
\end{aligned}
$$

where

$$
\mathcal{L}_d(\gamma) = -\log \Gamma \left(\textstyle\sum_{k=1}^{K} \gamma_k \right) + \sum_{k=1}^{K} \log \Gamma \gamma_k
$$



Write α
Write β
Parameters
Distributed Cache
Document
Document
Document
Document
Mapper: Update γ, φ
Mapper: Update γ, φ
Mapper: Update γ, φ
Mapper: Update γ, φ
Reducer
Reducer
Reducer
Sufficient Statistics for β Update
Hessian Terms
Driver: Update α
Test Likelihood Convergence

Map$(d, \vec{w})$

repeat
  for all $v \in [1, V]$ **do**
    for all $k \in [1, K]$ **do**
      Update $\phi_{v,k} = \beta_{v,k} \times \exp(\Psi(\gamma_{d,k}))$.
    end for
    Normalize row $\phi_{v,*}$, such that $\sum_{k=1}^{K} \phi_{v,k} = 1$.
    Update $\sigma = \sigma + \vec{w}_v \phi_v$, where ϕ_v is a K-dimensional vector, and $\vec{w}_v$ is the count of v in this document.
  end for
  Update row vector $\gamma_{d,*} = \alpha + \sigma$.
until convergence
for all $k \in [1, K]$ **do**
  for all $v \in [1, V]$ **do**
    Emit key-value pair $\langle k, \triangle \rangle : \vec{w}_v \phi_v$.
    Emit key-value pair $\langle k, v \rangle : \vec{w}_v \phi_v$. {order inversion}
  end for
  Emit key-value pair $\langle \triangle, k \rangle : (\Psi(\gamma_{d,k}) - \Psi(\sum_{l=1}^{K} \gamma_{d,l}))$. {emit the γ-tokens for α update}
  Output key-value pair $\langle k, d \rangle - \gamma_{d,k}$ to file.
end for
Emit key-value pair $\langle \triangle, \triangle \rangle - \mathcal{L}$, where $\mathcal{L}$ is log-likelihood of this document.

Input:

KEY - key pair $\langle p_{\text{left}}, p_{\text{right}} \rangle$.
VALUE - an iterator $\mathcal{I}$ over sequence of values.

Configuration

Initialize the total number of topics as K.
Initialize a normalization factor $n = 0$.

Reduce

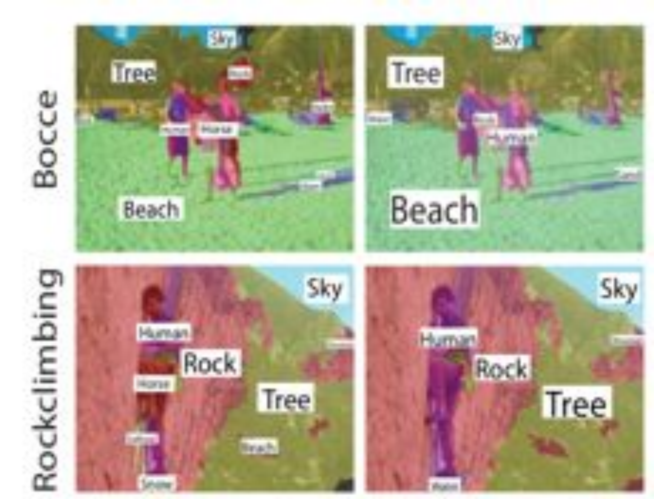
Compute the sum σ over all values in the sequence $\mathcal{I}$.
if $p_{\text{left}} = \triangle$ **then**
    if $p_{\text{right}} = \triangle$ **then**
        Output key-value pair $\langle \triangle, \triangle \rangle - \sigma$ to file.
        {output the model likelihood $\mathcal{L}$ for convergence checking}
    else
        Output key-value pair $\langle \triangle, p_{\text{right}} \rangle - \sigma$ to file.
        {output the γ-tokens to update α-vectors, Section **??**}
    end if
else
    if $p_{\text{right}} = \triangle$ **then**
        Update the normalize factor $n = \sigma$. {order inversion}
    else
        Output key-value pair $\langle k, v \rangle : \frac{\sigma}{n}$. {output normalized β value}
    end if
end if

Applicatons

- What's a document?
- What's a word?
- What's your vocabulary?
- How do you evaluate?

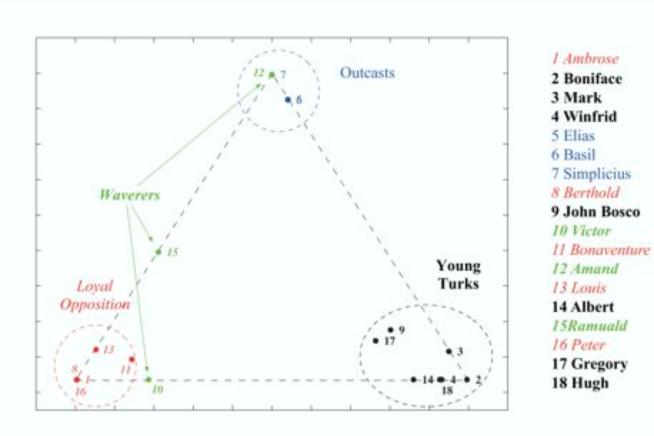
Applications

Computer Vision [*Li Fei-Fei and Perona* 2005]



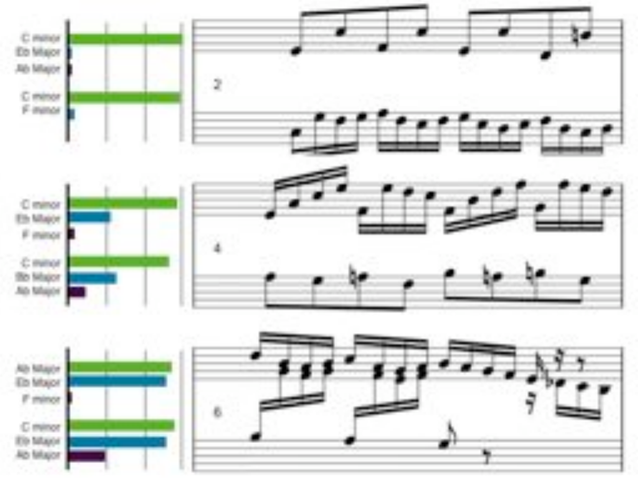
Applications

Social Networks [*Airoldi et al.* 2008]



Applications

Music [*Hu and Saul* 2009]



Nonparametric Models

- We've always assumed a fixed number of topics
- Topic modeling inspired resurgence of nonparametric Bayesian statistics that can handle infinitely many mixture components [*Antoniak* 1974]
- Equivalent to Rational Model of Categorization [*Griffiths et al.* 2007a]
- For the rest of this talk, cartoon version - details similar to LDA

Active research

- Combining these models with models of syntax
- Scaling up to larger corpora
- Making topics relevant to social scientists
- Humans in the loop
- Modeling metadata in document collections

Recap

- Probabilistic models - way of learning what data you have
- Make predictions about the future
- Require a little bit of math to figure out, but fairly easy to implement in MapReduce

Edoardo M. Airoldi, David M. Blei, Stephen E. Fienberg, and Eric P. Xing.
2008.
Mixed membership stochastic blockmodels.
J. Mach. Learn. Res., 9:1981–2014.

Charles E. Antoniak.
1974.
Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems.
The Annals of Statistics, 2(6):1152–1174.

David M. Blei, Andrew Ng, and Michael Jordan.
2003.
Latent Dirichlet allocation.
Journal of Machine Learning Research, 3:993–1022.

Jordan Boyd-Graber, David M. Blei, and Xiaojin Zhu.
2007.
A topic model for word sense disambiguation.
In *Proceedings of Emperical Methods in Natural Language Processing*.

Samuel Brody and Mirella Lapata.
2009.
Bayesian word sense induction.
In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 103–111, Athens, Greece, March. Association for Computational Linguistics.

T. L. Griffiths, K. R. Canini, A. N. Sanborn, and D. J. Navarro.
2007a.
Unifying rational models of categorization via the hierarchical Dirichlet process.
In *Proceedings of the Twenty-Ninth Annual Conference of the Cognitive Science Society*.

Thomas L. Griffiths, Mark Steyvers, and Joshua Tenenbaum