# CS 589 Fall 2021 Lecture 6

## Topic model

**Monday 6:30-9:00**
**Babbio 122**

All zoom links in Canvas

photo: https://www.scubedstudios.com/information-retrieval/

# Review of Lecture 5

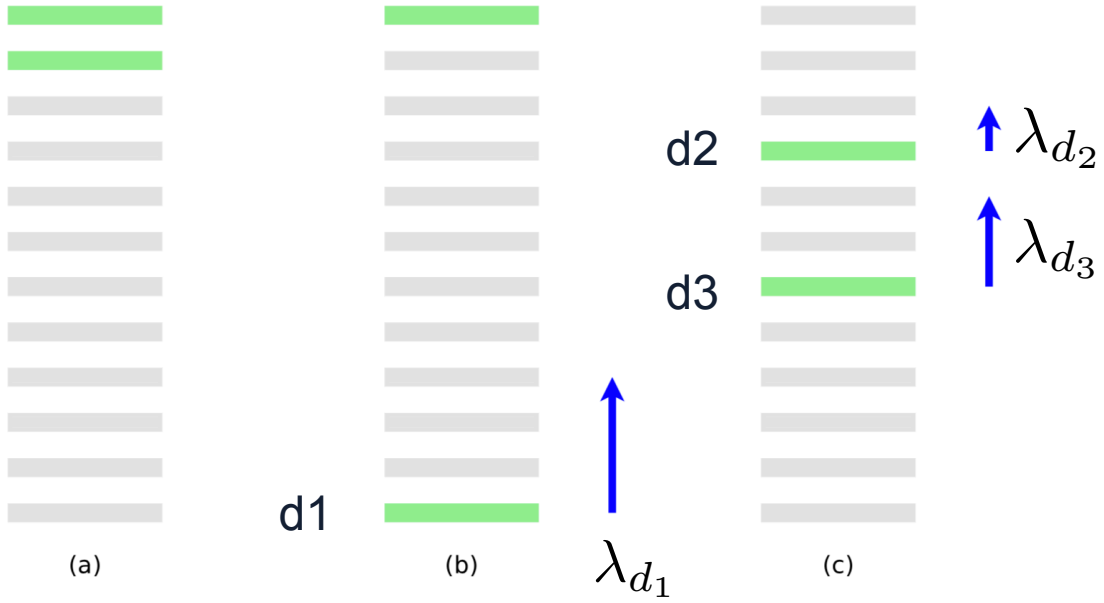
- The idea of machine learning for ranking has been around for a long time
- Learning to rank can be turned into a classification or regression problem
- LTR algorithms:
  - Multiple additive regression tree (MART)
  - RankNet
- LTR categories:
  - Pointwise
  - Pairwise
  - Listwise
- Yahoo! Learning to rank challenge (Tree ensemble algorithm wins!)

# Review of Lecture 5

- Multiple additive regression tree (MART)
  - Minimizes the square loss (i.e., just like linear regression and regression tree)
  - First iteration: f0(x) = avg of all y_i
  - From iteration 1 through k:
    - Update the residual of each y_i
    - Fit a regression tree fk(x) for the residuals
    - Update the sum F(x)
- The name “gradient boosting regression tree” comes from the fact that the residual = gradient of the objective function

# Review of Lecture 5

- RankNet
  - Convert the pairwise loss function to the summation of $\lambda_i$ for each document di
  - That is, $\lambda_i$ is a sort of "gradient" with respect to di




$$w_k = w_k - \alpha \frac{\partial \Sigma}{\partial w_k}$$

$$\frac{\partial \Sigma}{\partial w_k} = \uparrow \cdot \frac{\partial s_1}{\partial w_k} + \uparrow \cdot \frac{\partial s_2}{\partial w_k} + \uparrow \cdot \frac{\partial s_3}{\partial w_k}$$

$$s_i = wx_i + b$$

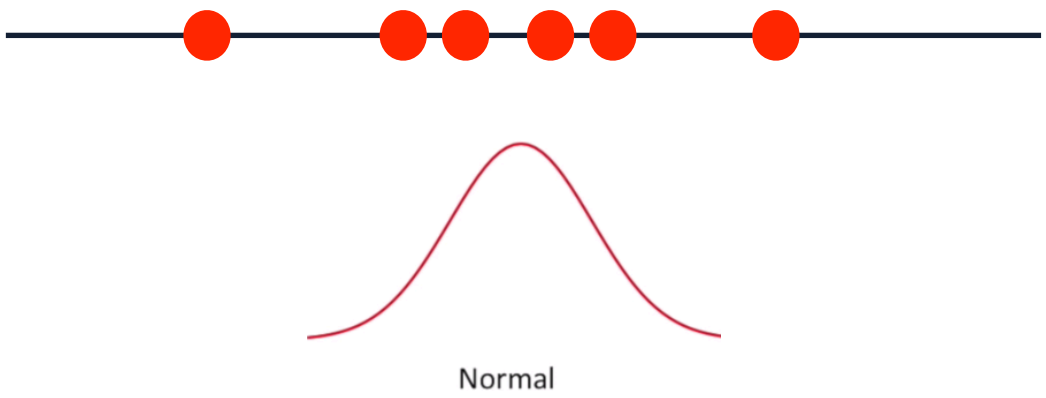
# Lecture 6: Topic model

- Pop quiz
- Expectation maximization
- Probabilistic topic model
  - The toy coin topic problem
  - Probabilistic latent semantic analysis
- Latent Dirichlet allocation
  - Bayesian inference of topic model
  - Variational inference for LDA
  - Gibbs sampling, Markov chain Monte-Carlo

# Maximum likelihood estimation

- Estimating the underlying distribution behind the observations
  - i.e., by building a model with parameter $\theta$ and find the $\theta$ that maximizes the likelihood of the observation

- Example 1: Estimating the weights of mice from observations x1,...,xn

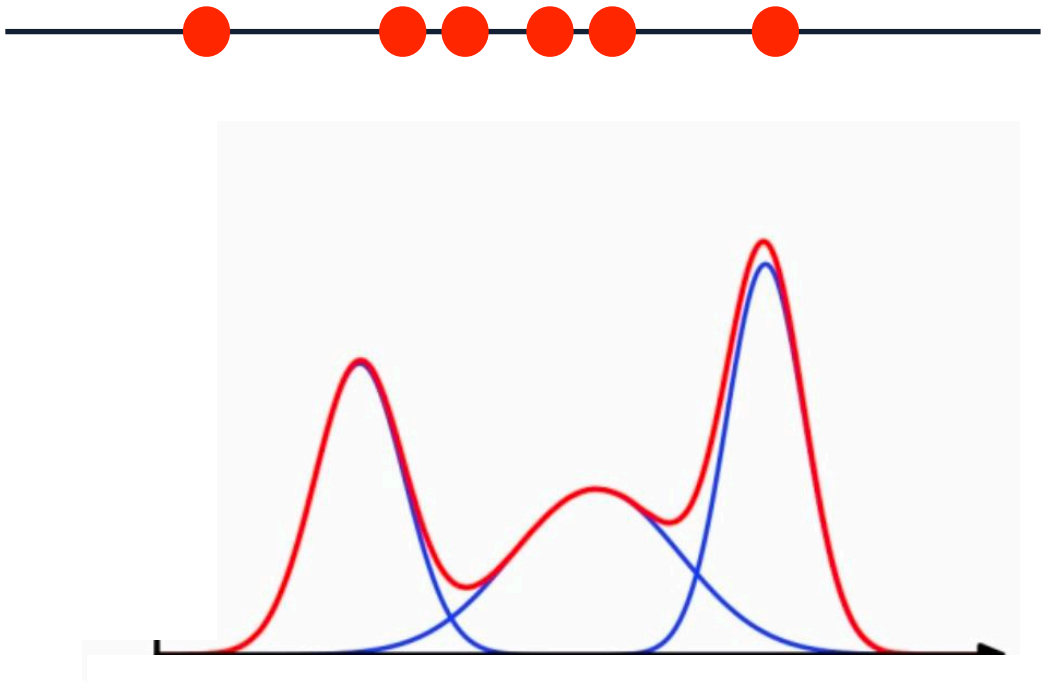



$$\theta^* = argmax_{\mu,\sigma} \sum_{i=1}^{n} f_X(x_i; \mu, \sigma^2)$$

$$= argmax_{\mu,\sigma} \sum_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}$$

# Maximum likelihood estimation for mixture models

- Example 2: Estimating the weights of mice from observations x1,...,xn



$$\theta^* = argmax_{\mu,\sigma} \sum_{i=1}^{n} \sum_{t=1}^{3} p(z_i = t) \cdot f_X(x_i; \mu_t, \sigma_t^2)$$

$$= argmax_{\mu,\sigma} \sum_{i=1}^{n} \sum_{t=1}^{3} p(z_i = t) \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x_i-\mu_t)^2}{2\sigma_t^2}}$$

**latent variable zi: which topic xi comes from**

# Expectation Maximization

- The expectation maximization algorithm is an algorithm for maximum likelihood estimation under the ***latent space*** assumption
  - Likelihood of the incomplete space: the observation

$$p(\mathbf{x}; \theta)$$

  - Likelihood of the complete space: the observation + the latent variable, e.g., the latent topic

$$p(\mathbf{x}, z|\theta)$$

  - Estimating the incomplete probability by marginalizing out the complete space

$$p(\mathbf{x}; \theta) = \sum_{k=1}^{K} p(\mathbf{x}|z; \theta)p(z = k)$$

# Expectation maximization algorithm

- EM algorithm: repeat n=1…N:

**E step:** $$Q(\theta|\hat{\theta}^{(n)}) = \mathbb{E}_{z|\mathbf{x},\hat{\theta}^{(n)}}[\log p(\mathbf{x}, z|\theta)]$$

**M step:** $$\hat{\theta}^{(n+1)} = arg\max_{\theta \in \mathcal{S}} Q(\theta|\hat{\theta}^{(n)})$$

- **Theorem**: the likelihood of observation, $\log p(\mathbf{x}; \theta^{(n)})$ , monotonously increases with *n (Proof on whiteboard)*

# An example problem: Coin-topic problem

- Author H and author T are co-authoring a paper in the following way:
  - At each time, they toss a coin to write the next word. If it's "head", author H writes the next word, if it's "tail", author T writes the next word. The probability for "head" is $\lambda$
  - The head author selects the next word by randomly sampling from $p(w|H)$ , the tail author selects the next word by randomly sampling from $p(w|T)$

# Problem 1: $p(w|H)$ and $p(w|T)$ are known, estimate $\lambda$

$$l_{cd}(\theta) = \sum_i \log p(w_i|z_i)p(z_i)$$

$$\mathbb{E}_{z|d,\lambda}[l_{cd}(\theta)] = \max_{\lambda} \sum_i p(z_i = H|d,\lambda) \log p(w_i|H)\lambda + p(z_i = T|d,\lambda) \log p(w_i|T)(1-\lambda)$$

$$= \sum_i \sum_v \mathbb{1}\left[w_i = v\right](p(z = H|v) \log p(v|H)\lambda + p(z = T|v) \log p(v|T)(1-\lambda))$$

- where both head and tail distributions are known, e.g.:

| | the | computer | data | baseball | game | interesting |
|---|---|---|---|---|---|---|
| p(w\|T) | 0.2 | 0.05 | 0.05 | 0.4 | 0.2 | 0.1 |
| p(w\|H) | 0.25 | 0.2 | 0.2 | 0.15 | 0.1 | 0.1 |

# Expectation maximization

- Find $\theta$ that maximizes $\mathbb{E}_{z|d,\lambda}[l_{cd}(\theta)]$ (take the derivative):

$$\sum_i \sum_v \mathbb{1}[w_i = v](p(z = H|v)\frac{1}{\lambda} + p(z = T|v)\frac{1}{1-\lambda} = 0$$

$$\Rightarrow \lambda \propto \sum_v p(z = H|v)(\sum_i \mathbb{1}[w_i = v])$$

$$\Rightarrow \lambda^{(n+1)} \propto \frac{1}{|d|}\sum_v p^{(n)}(z = H|v)count(d, v)$$ **(M step)**

# Expectation maximization

- E step: apply Bayes' theorem:

$$p^{(n+1)}(z = H|v) \propto p(v|z = H)p(z = H) = p(v|H)\lambda^{(n)}$$

$$p^{(n+1)}(z = T|v) \propto p(v|z = T)p(z = T) = p(v|T)(1 - \lambda^{(n)})$$

# Problem 2: $\lambda$ and $p(w|H)$ known, estimating $p(w|T)$

$$\mathbb{E}_{z|d,\lambda}[l_{cd}(\theta)] = \max_{\lambda} \sum_i p(z_i = H|d,\lambda) \log p(w_i|H)\lambda + p(z_i = T|d,\lambda) \log p(w_i|T)(1-\lambda)$$

$$-\eta \left( \sum_v p(v|T) - 1 \right)$$

$$= \sum_i \sum_v \mathbb{1}\left[w_i = v\right] p(z = H|v) \log p(v|H)\lambda + p(z = T|v) \log p(v|T)(1-\lambda)$$

$$-\eta \left( \sum_v p(v|T) - 1 \right)$$

- Take the derivative for $p(v|T)$ and set to 0, we can get (M step):

$$p^{(n+1)}(v|T) = \frac{\sum_i \mathbb{1}\left[w_i = v\right] p^{(n)}(z = H|v)}{\sum_v \sum_i \mathbb{1}\left[w_i = v\right] p^{(n)}(z = H|v)}$$

# Problem 2: $\lambda$ and $p(w|H)$ known, estimating $p(w|T)$

$$\mathbb{E}_{z|d,\lambda}[l_{cd}(\theta)] = \max_{\lambda} \sum_i p(z_i = H|d,\lambda) \log p(w_i|H)\lambda + p(z_i = T|d,\lambda) \log p(w_i|T)(1-\lambda)$$

$$-\eta \left( \sum_v p(v|T) - 1 \right)$$

$$= \sum_i \sum_v \mathbb{1}\left[w_i = v\right] p(z = H|v) \log p(v|H)\lambda + p(z = T|v) \log p(v|T)(1-\lambda)$$

$$-\eta \left( \sum_v p(v|T) - 1 \right)$$

- Take the derivative for $p(v|T)$ and set to 0, we can get (M step):

$$p^{(n+1)}(v|T) = \frac{\sum_i \mathbb{1}\left[w_i = v\right] p^{(n)}(z = H|v)}{\sum_v \sum_i \mathbb{1}\left[w_i = v\right] p^{(n)}(z = H|v)}$$

**(E step same as Problem 1)**

# Problem 2: $\lambda$ and $p(w|H)$ known, estimating $p(w|T)$

- How can Problem 2 be useful?
  - Suppose p(w|T) is the main topic (Python)
  - p(w|H) is the background topic (StackOverflow general question)
  - The mixture of head and tail topic is dominated by background words:
- After stop words removal, the true topic p(w|T) is “revealed”:

# Problem 3: Unknown topic and $\lambda$

- Suppose p(w|H) is unknown, but p(w|T) and lambda are both unknown:
  - Use Problem 1 to estimate lambda
  - Use Problem 2 to estimate p(w|T)
  - Same E step

# Applications of Coin Topic Problem for Text Mining

- Application Scenarios:
  - p(w|H) & p(w|T) are known; estimate λ
  - p(w|H) & λ are known; estimate p(w|T)
  - p(w|H) is known; estimate λ & p(w|T)
  - λ is known; estimate p(w|H)& p(w|T)
  - Estimate λ, p(w|H), p(w|T)

how much percent of the document is about computer game?

30% of the doc is about computer game, what's the other topic about?

The doc is about computer game, is it also about some other topic, and if so to what extent?

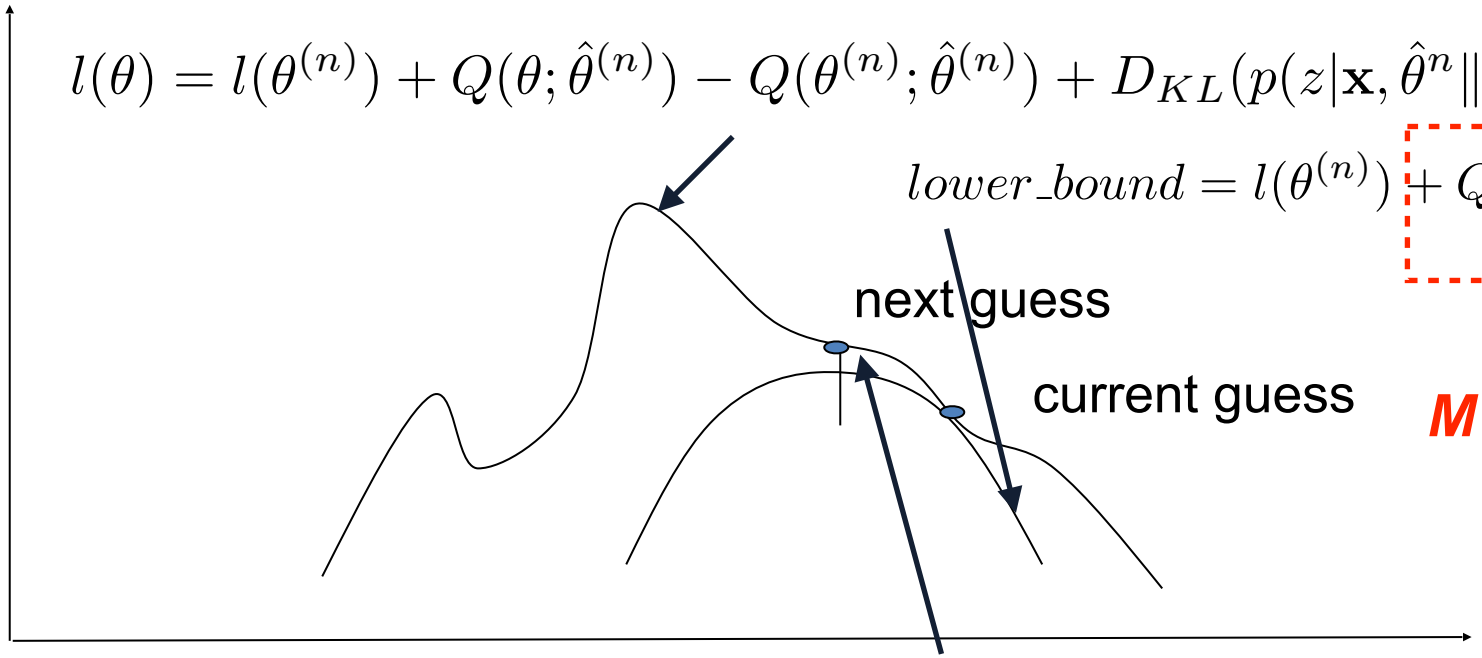
30% of the doc is about one topic and 70% is about another, what are these two topics?

The doc is about two subtopics, find out what these two subtopics are and to what extent the doc covers each.

# Expectation maximization as hill climbing

$$l(\theta) = l(\theta^{(n)}) + Q(\theta; \hat{\theta}^{(n)}) - Q(\theta^{(n)}; \hat{\theta}^{(n)}) + D_{KL}(p(z|\mathbf{x}, \hat{\theta}^n \| p(z|\mathbf{x}, \hat{\theta}))$$

$$lower\_bound = l(\theta^{(n)}) + Q(\theta; \hat{\theta}^{(n)}) - Q(\theta^{(n)}; \hat{\theta}^{(n)})$$




***M step object***

***converge to local optimal***

***Slides from UIUC CS510***

# EM algorithm in action

- Log likelihood increases:

| Word | # | $p(w\|\theta_B)$ | Iteration 1 | | Iteration 2 | | Iteration 3 | |
|---|---|---|---|---|---|---|---|---|
| | | | $P(w\|\theta)$ | $p(z=0\|w)$ | $P(w\|\theta)$ | $P(z=0\|w)$ | $P(w\|\theta)$ | $P(z=0\|w)$ |
| The | 4 | 0.5 | **0.25** | 0.33 | **0.20** | 0.29 | **0.18** | 0.26 |
| Paper | 2 | 0.3 | **0.25** | 0.45 | **0.14** | 0.32 | **0.10** | 0.25 |
| Text | 4 | 0.1 | **0.25** | 0.71 | **0.44** | 0.81 | **0.50** | 0.93 |
| Mining | 2 | 0.1 | **0.25** | 0.71 | **0.22** | 0.69 | **0.22** | 0.69 |
| Log-Likelihood | | | -16.96 | | -16.13 | | -16.02 | |

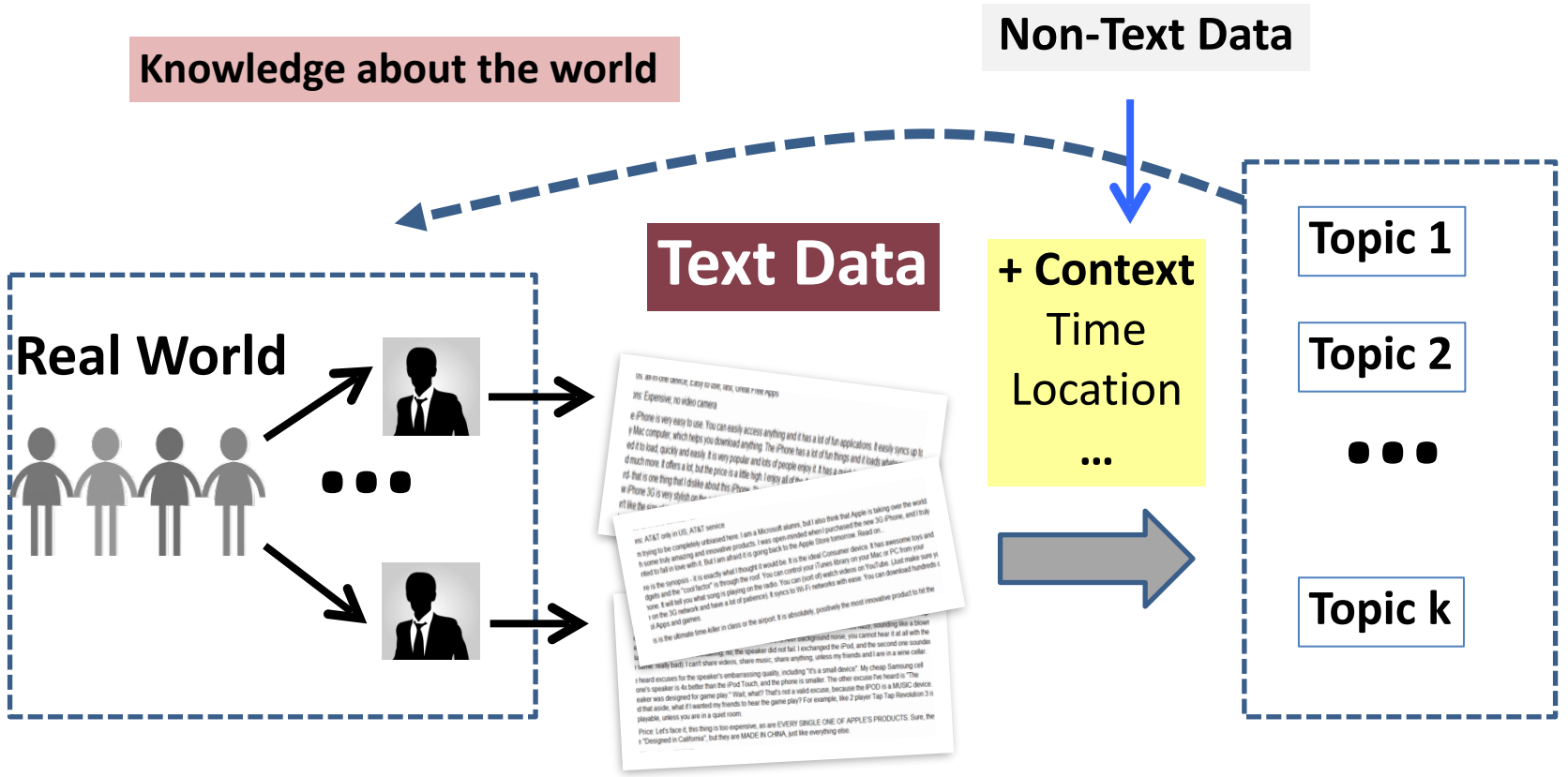
# Topic models and analysis

- Topic ≈ main idea discussed in text data
  - Theme/subject of a discussion or conversation
  - Different granularities (e.g., topic of a sentence, an article, etc.)

- Many applications require discovery of topics in text
  - What are Twitter users talking about today?
  - What are the current research topics in data mining? How are they different from those 5 years ago?
  - What do people like about the iPhone 6? What do they dislike?
  - What were the major topics debated in 2012 presidential election?

*Slides from UIUC CS510*

# Lifecycle of topic and text data

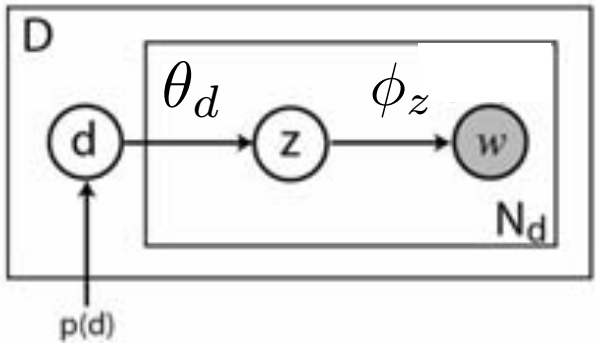



*Slides from UIUC CS510*

# Probabilistic latent semantic analysis

- Assume documents are generated by sampling words from k latent topics

- For each document d:
  - For each token position i
  - Choose a topic z ~ Multinomial( $\theta_d$ )
  - Choose a term w ~ Multinomial( $\phi_z$ )

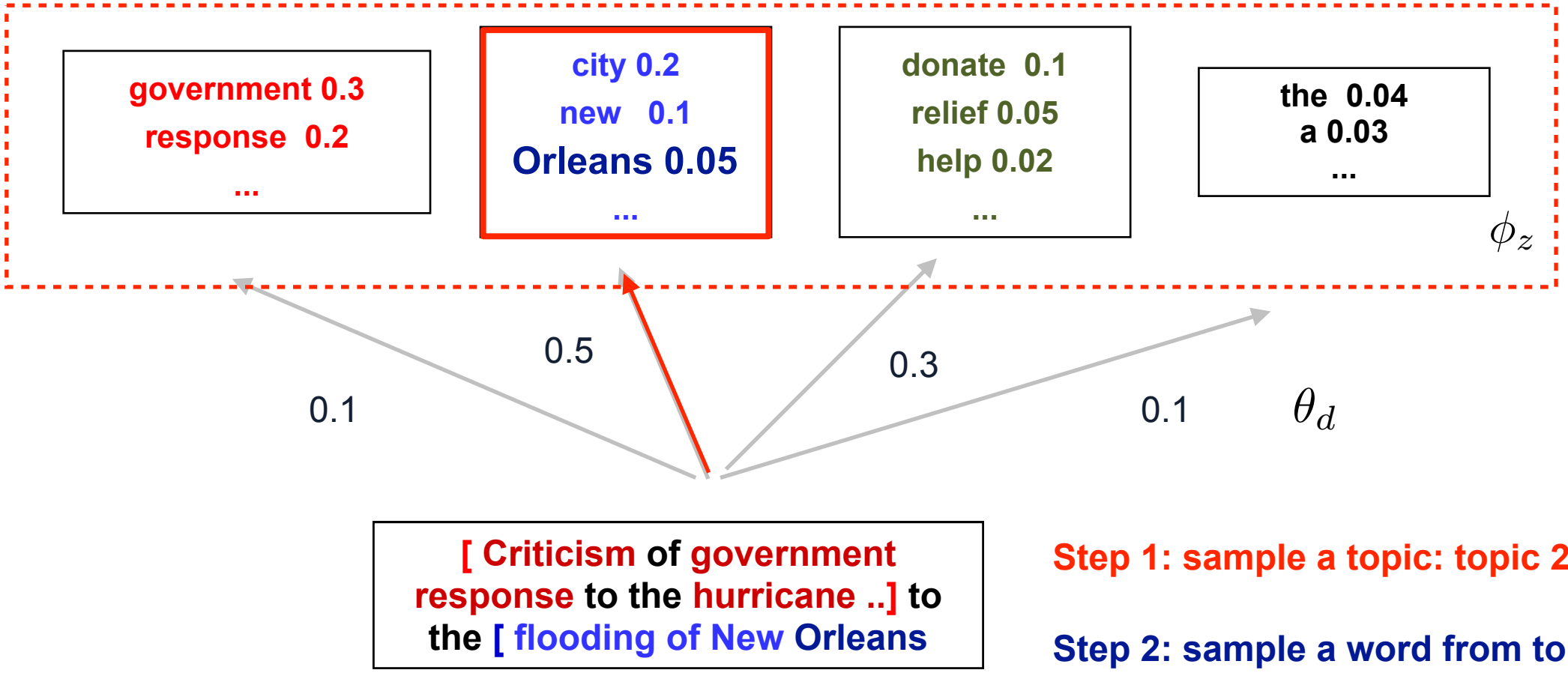



*Probabilistic Latent Semantic Analysis. Thomas Hoffman. 2001.*

# Review of PLSA

Phi T x V, V: vocabulary size ~50,000, T: #topics, T=20

theta D x T, D: #documents: ~10,000, T: #topics, T=20

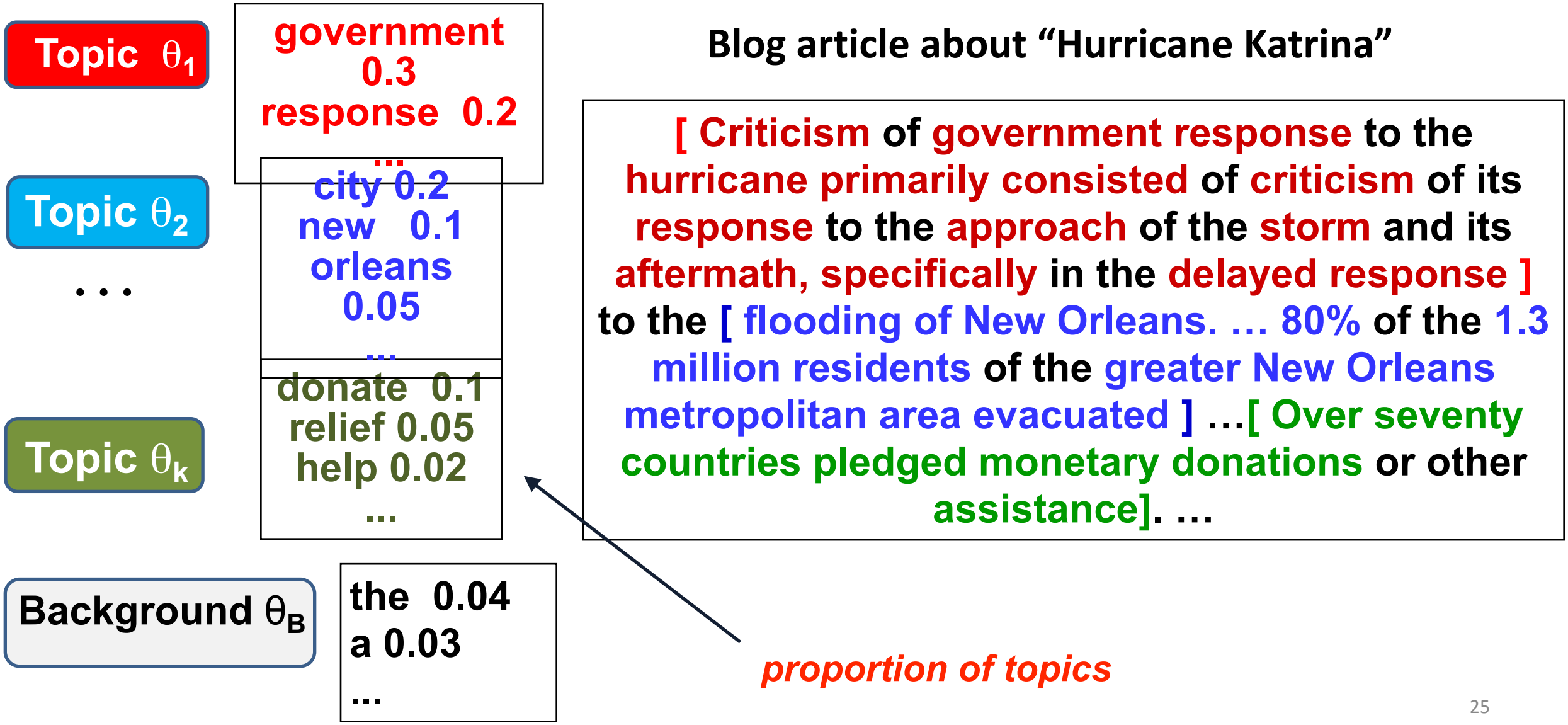



**Step 1: sample a topic: topic 2**

**Step 2: sample a word from topic 2**

*Probabilistic Latent Semantic Analysis. Thomas Hoffman. 2001.*

# Document as a Sample of Mixed Topics



# Probabilistic latent semantic analysis

$$p(d_i = w|\Phi, \theta_d) = \sum_{z=1}^{T} \phi_{z,w}\theta_{d,z}$$

$$p(\mathcal{W}|\Phi, \Theta)$$
$$= \prod_{d=1}^{D} \prod_{d_i=1}^{N_d} \sum_{z=1}^{T} \phi_{z,w}\theta_{d,z}$$
$$= \prod_{d=1}^{D} \prod_{w=1}^{V} (\sum_{z=1}^{T} \phi_{z,w}\theta_{d,z})^{count(d,w)}$$

$$arg\max_{\Phi,\Theta}[\log p(\mathcal{W}|\Phi, \Theta) + \sum_{d=1}^{D} \lambda_d(1 - \sum_{z=1}^{T} \theta_{(d,z)}) + \sum_{z=1}^{T} \sigma_k(1 - \sum_{w=1}^{V} \phi_{z,w})]$$

# Probabilistic latent semantic analysis

$$argmax_{\Theta,\Phi} \sum_d \sum_v count(d,v) \sum_t^T R_{d,v,t}(\log \phi_{t,v} + \log \theta_{d,t})$$
$$+ \left( \sum_d \lambda_d \left( 1 - \sum_t^T \theta_{d,t} \right) + \sum_t^T \sigma_t \left( 1 - \sum_v \phi_{t,v} \right) \right)$$

$R_{d,v,t}$ : latent variable's conditional probability, i.e., $p(t|v,d)$

- **M step: take the derivative of the above formula for $\theta_{d,t}, \phi_{t,v}$ :**

$$\theta_{d,t} \propto \sum_v R_{d,v,t} count(d,v) \qquad \Rightarrow \theta_{d,t} = \frac{\sum_v R_{d,v,t} count(d,v)}{\sum_t \sum_v R_{d,v,t} count(d,v)}$$

$$\phi_{t,v} \propto \sum_d R_{d,v,t} count(d,v) \qquad \Rightarrow \phi_{t,v} = \frac{\sum_d R_{d,v,t} count(d,v)}{\sum_v \sum_d R_{d,v,t} count(d,v)}$$

- **E step: apply Bayes theorem:** $R_{d,v,t} \propto \phi_{d,t}\theta_{t,v} \Rightarrow R_{d,v,t} = \frac{\phi_{d,t}\theta_{t,v}}{\sum_t \phi_{d,t}\theta_{t,v}}$

# Probabilistic latent semantic analysis: partially available labels

- Generalized topic modeling:
  - Each document can contain just one topic, e.g., short documents
  - That is, topic inference = topic classification

- If we already know the document tags for a part of the documents, does the partial labels help us make better predictions for the entire corpus? **(Homework 3)**

- Example: news tagging, StackOverflow question tagging

# Probabilistic latent semantic analysis: partially available labels

**Trends for you** · Change

**#RAF100**
The RAF celebrates 100th anniversary

**#TuesdayThoughts**
@NWMCblog is Tweeting about this

**#ThailandCaveRescue**
237K Tweets

**Thai Navy Seal**
120K Tweets

**All 12**
All 12 boys and coach rescued from Thai cave

**#IgniteB2B**
1,850 Tweets

**#WildBoars**
4,934 Tweets

**Science**
159K Tweets

**George Clooney**
George Clooney injured in motorcycle accident in Italy

**#NationalPinaColadaDay**
1,870 Tweets

# Homework 3

- Suppose each document has only 1 topic. We have two document set: S1 (100 documents) contains all the tagged documents; S2 (10,000 documents) contains all the untagged documents. Each tag is a topic, there are only 2 topics

- (Part 1): derive the EM algorithm using pLSA that maximizes the probability of the observed document, given the known topics from S1

- (Part 2): implement your EM algorithm, output the predicted topic for each document in S2

*Slides from UIUC CS510*

# CS 589 Fall 2021 Lecture 6

## Latent Dirichlet allocation

**Monday 6:30-9:00**
**Babbio 122**

All zoom links in Canvas

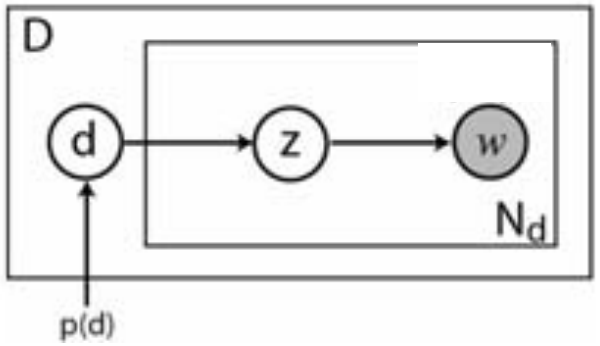
photo: https://www.scubedstudios.com/information-retrieval/

# Latent Dirichlet allocation [Blei et al. 03]

- A Bayesian topic model by considering the prior distribution of p(t|d)
  - pLSA assumes $\theta_{d,t}$ is a variable by itself
  - LDA assumes $\theta_{d,t}$ is drawn from yet another distribution (i.e., "distribution of distribution")

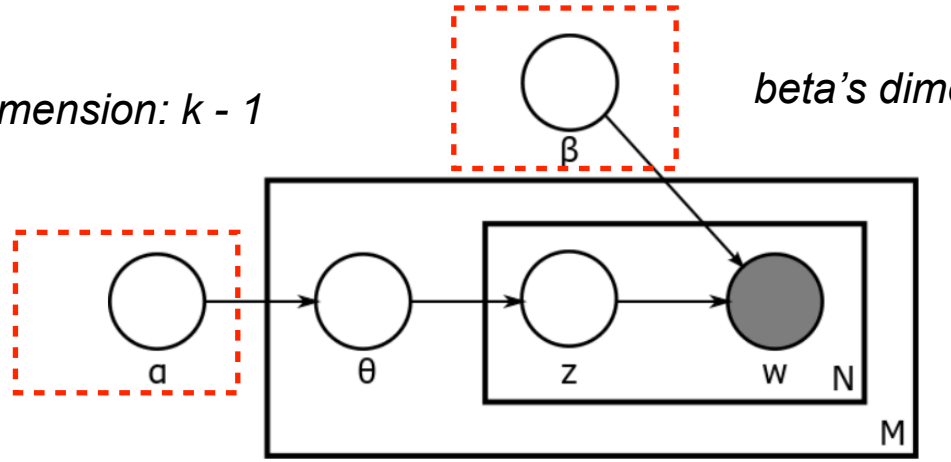
# Latent Dirichlet allocation [Blei et al. 2003]

- Adding a prior *hyper parameter* $\alpha$ to PLSA model
  - The document-topic probability is **randomly sampled** from the same prior distribution
  - That is, when computing the probability, we include $p(\theta_{d,t}|\alpha)$ in the model




*alpha's dimension: k - 1*

*beta's dimension: V - 1*



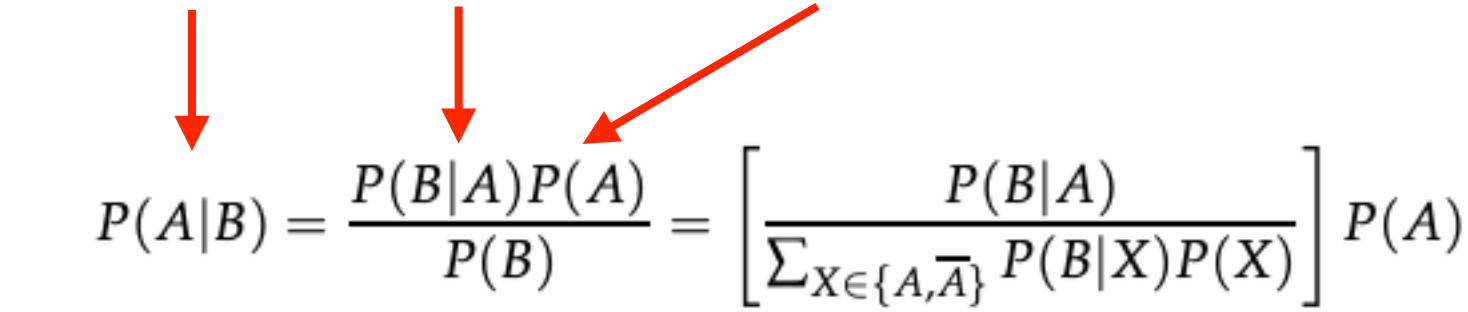
# Bayes' rules

**Chain rule:** joint distribution

$$P(A,B) = P(A \cap B) = P(A|B)P(B) = P(B|A)P(A)$$

**Bayes' rule:**

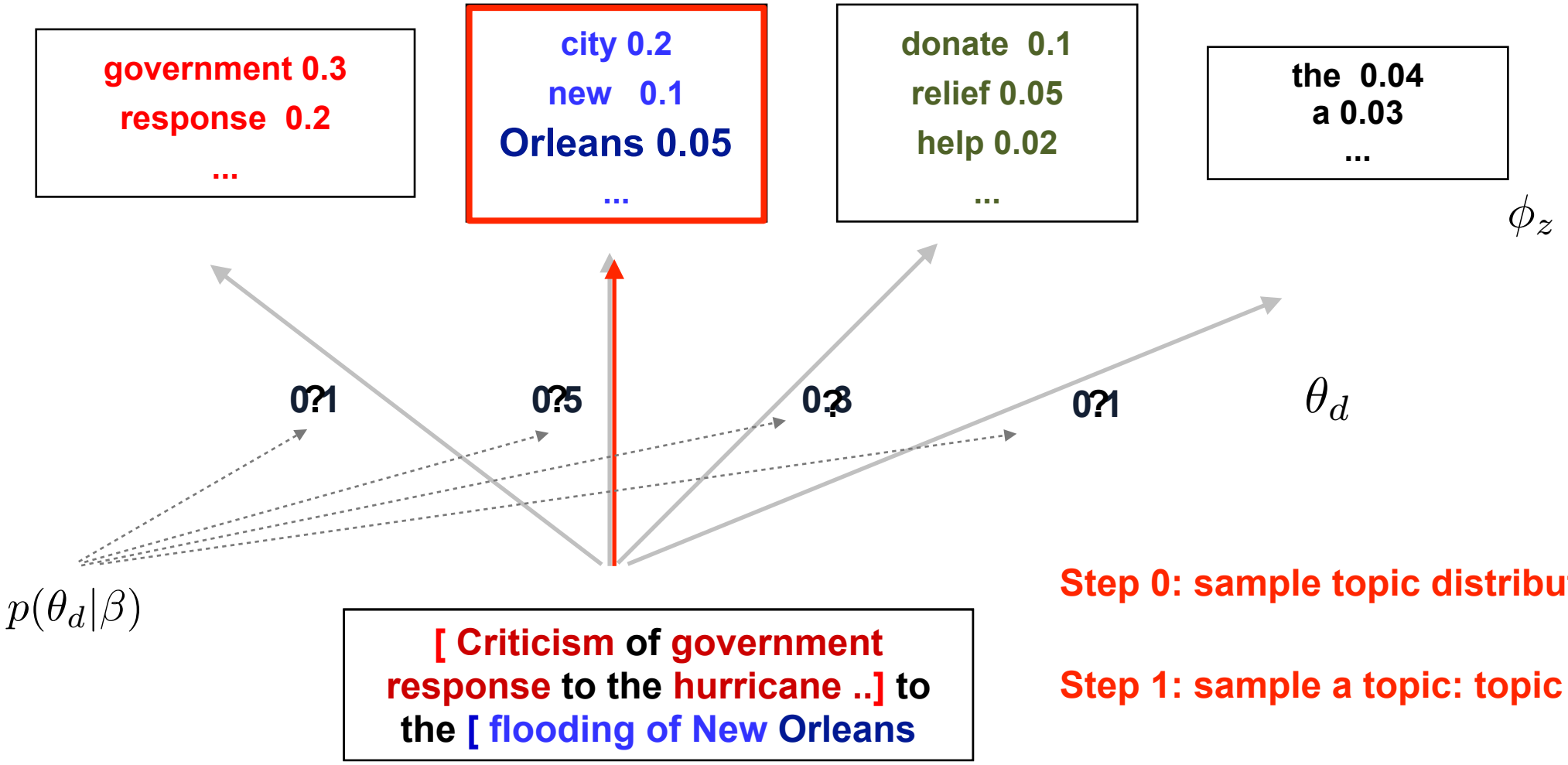
posterior likelihood prior



$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} = \left[\frac{P(B|A)}{\sum_{X\in\{A,\overline{A}\}} P(B|X)P(X)}\right] P(A)$$

$$P(A|B) \propto P(B|A)P(A)$$

# PLSA -> LDA




Step 0: sample topic distribution

Step 1: sample a topic: topic 2

Step 2: sample a word from topic 2

# Maximum a Posterior (MAP) estimation

- Maximum likelihood estimation:

$$\theta_{MLE} = arg\max_{\theta} p(x_1, \cdots, x_n|\theta)$$

- Choose the parameter with the maximum posterior probability

$$\begin{aligned}\theta_{MAP} &= arg\max_{\theta} p(\theta|x_1, \cdots, x_n) \\ &= arg\max_{\theta} p(x_1, \cdots, x_n|\theta)p(\theta)\end{aligned}$$

- LDA is the MAP version of pLSA (MLE)

# PLSA -> LDA

- PLSA:

$$p_d\left(w \mid \{\theta_d\}, \{\phi_z\}\right) = \sum_z \theta_{d,z}\phi_{z,w}$$

$$\log p\left(D \mid \{\phi_z\}, \{\theta_d\}\right) = \sum_{d \in D}\sum_{w \in V} c(w,d) \log \left[\sum_z \theta_{d,z}\theta_{z,w}\right]$$

- LDA:

$$p_d\left(w \mid \{\theta_d\}, \{\phi_z\}\right) = \sum_z \theta_{d,z}\phi_{z,w}$$

$$\log p\left(d \mid \alpha, \{\phi_z\}\right) = \int \sum_{w \in V} c(w,d) \log \left[\sum_z \theta_{d,z}\phi_{z,w}\right] p\left(\theta_d \mid \alpha\right) d\theta_d$$

**marginalized probability:**

$$\log p(D \mid \alpha, \beta) = \int \sum_{d \in D} \log p\left(d \mid \alpha, \{\phi_z\}\right) \prod_z p\left(\phi_z \mid \beta\right) d\phi_1 \ldots d\phi_k$$

# Solving Maximum a Posteriori inference for LDA

- Exact inference is intractable:

$$p(Z, \Phi, \Theta | D, \alpha, \beta) = \frac{p(D, Z, \Phi, \Theta | \alpha, \beta)}{p(D | \alpha, \beta)}$$

$$\log p(D \mid \alpha, \beta) = \int \sum_{d \in D} \log p\left(d \mid \alpha, \{\phi_z\}\right) \prod_{z=1}^{k} p\left(\phi_z \mid \beta\right) d\phi_1 \ldots d\phi_k$$

- Equation (1) is computationally intractable due to the coupling of beta and phi in the denominator

# Variational inference

- Key idea: use a **surrogate distribution** to approximate the posterior distribution of latent variables
  - Surrogate distribution is simpler to estimate than the true posterior

- Goal: finding the “best” surrogate distribution from a certain parametric family by **minimizing** the KL-divergence between the surrogate function (Q) to the true posterior (P)

# Evidence lower bound (ELBO)

- Given that $p(Z|D) = \frac{p(D,Z)}{p(D)}$ , we want to minimize the KL divergence between Q(Z) and p(Z|D):

$$
\begin{aligned}
D_{\mathrm{KL}}(q\|p) &= \int_Z q(Z)\left[\log\frac{q(Z)}{p(Z,D)} + \log p(D)\right] \\
&= \int_Z q(Z)[\log q(Z) - \log p(Z,D)] + \int_Z q(Z)[\log p(D)] \\
&= \int_Z q(Z)[\log q(Z) - \log p(Z,D)] + \log p(D)
\end{aligned}
$$

$$
\Rightarrow \log p(D) = D_{\mathrm{KL}}(q\|p) - \mathbb{E}_{\mathrm{q}}[\log q(Z) - \log p(Z,D)] = D_{\mathrm{KL}}(q\|p) + \mathcal{L}(q)
$$

# Evidence lower bound (ELBO)

- p(D) does not rely on the latent variable Z:

$$\Rightarrow \log p(D) = D_{\mathrm{KL}}(q\|p) - \mathbb{E}_{\mathrm{q}}[\log q(Z) - \log p(Z, D)] = D_{\mathrm{KL}}(q\|p) + \mathcal{L}(q)$$

- Therefore minimizing KL divergence = maximizing L(q), which we call evidence lower bound, as KL divergence is non-negative

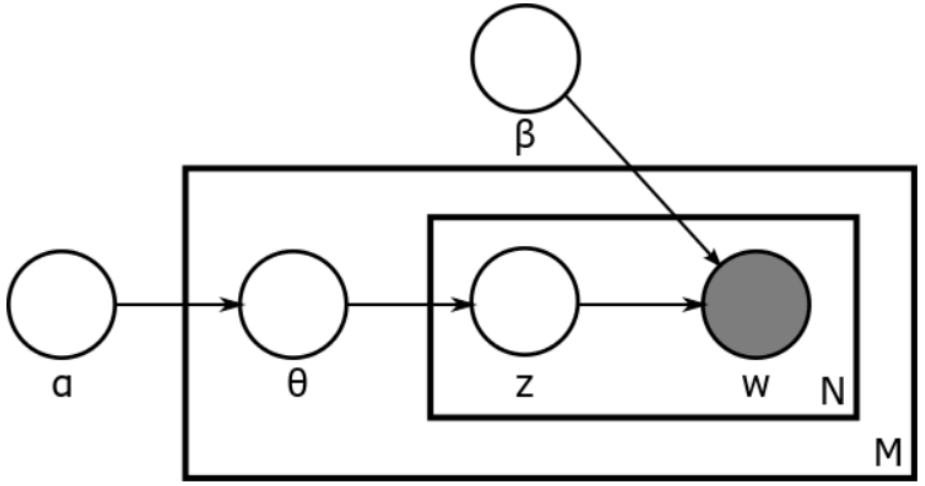
- The lower bound is more tractable to optimize than the evidence:

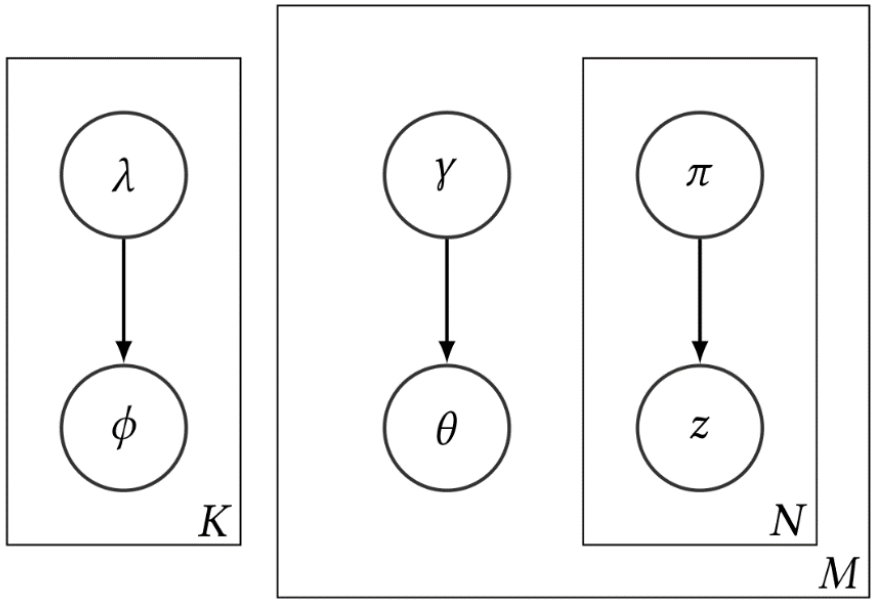
$$L(q) = \int_Z q(Z)[\log q(Z) - \log p(Z, D)]$$

# LDA variational inference

- Assumption: q belongs to a family of distribution which can be factorized

$$q(\theta, \mathbf{z} \mid \gamma, \phi) = q(\theta \mid \gamma) \prod_{n=1}^{N} q\left(z_n \mid \phi_n\right)$$



# LDA variational inference

$$L(q) = \int_Z q(Z)[\log q(Z) - \log p(Z, D)]$$

$$p(Z, \Phi, \Theta|D, \alpha, \beta) = \frac{p(D, Z, \Phi, \Theta|\alpha, \beta)}{p(D|\alpha, \beta)}$$

- ELBO of LDA under the factorization assumption:

$$L(\gamma, \phi; \alpha, \beta) = \mathrm{E}_q[\log p(\theta \mid \alpha)] + \mathrm{E}_q[\log p(\mathbf{z} \mid \theta)] + \mathrm{E}_q[\log p(\mathbf{w} \mid \mathbf{z}, \beta)] - \mathrm{E}_q[\log q(\theta)] - \mathrm{E}_q[\log q(\mathbf{z})]$$

$$\begin{aligned} L(\gamma, \phi; \alpha, \beta) = {} & \log \Gamma\left(\sum_{j=1}^{k} \alpha_j\right) - \sum_{i=1}^{k} \log \Gamma(\alpha_i) + \sum_{i=1}^{k} (\alpha_i - 1)\left(\Psi(\gamma_i) - \Psi\left(\sum_{j=1}^{k} \gamma_j\right)\right) \\ & + \sum_{n=1}^{N} \sum_{i=1}^{k} \phi_{ni}\left(\Psi(\gamma_i) - \Psi\left(\sum_{j=1}^{k} \gamma_j\right)\right) \\ & + \sum_{n=1}^{N} \sum_{i=1}^{k} \sum_{j=1}^{V} \phi_{ni} w_n^j \log \beta_{ij} \\ & - \log \Gamma\left(\sum_{j=1}^{k} \gamma_j\right) + \sum_{i=1}^{k} \log \Gamma(\gamma_i) - \sum_{i=1}^{k} (\gamma_i - 1)\left(\Psi(\gamma_i) - \Psi\left(\sum_{j=1}^{k} \gamma_j\right)\right) \\ & - \sum_{n=1}^{N} \sum_{i=1}^{k} \phi_{ni} \log \phi_{ni} \end{aligned}$$

*$\theta$ and $z$ integrated out*

# LDA variational inference

- LDA variational inference algorithm: optimizing γ, φ, β

1. Randomly initialize variational parameters (can't be uniform)
2. For each iteration:
    1. For each document, update γ and φ
    2. For corpus, update β
    3. Compute L for diagnostics
3. Return **expectation of variational parameters** for solution to latent variables

$$\phi_{ni} \propto \beta_{iv} \exp\left(\psi\left(\gamma_i\right) - \Psi\left(\sum_j \gamma_j\right)\right)$$

$$\mathrm{E}_q\left[\log\left(\theta_i\right) \mid \gamma\right] = \Psi\left(\gamma_i\right) - \Psi\left(\sum_{j=1}^{k} \gamma_j\right)$$

# Pros and Cons of Variational inference

- Pros
  - Deterministic algorithm - easy to tell when converged
  - Embarrassingly parallelizable

- Cons:
  - Speed: make many calls to transcendental functions (no close form solution)
  - Quality is questionable due to the factorization assumption
  - Memory usage: requires O(MNK) to store the per-token variational distributions

# Conjugate prior distribution

- That is, the **posterior distributions** are in the same probability distribution family as the prior probability distribution
- The prior is called the **conjugate prior** of the likelihood probability
- Conjugate prior makes denominator easy to compute because you can integrate out theta
- e.g., Beta distribution is the conjugate prior of Bernoulli distribution

**Bernoulli:** $p(x \mid \theta) = \theta^x (1-\theta)^{1-x}.$

**Beta:** $p(\theta \mid \alpha) = K(\alpha)\theta^{\alpha_1 - 1}(1-\theta)^{\alpha_2 - 1}.$

$$\begin{aligned} K(\alpha) &= \left( \int \theta^{\alpha_1 - 1}(1-\theta)^{\alpha_2 - 1} d\theta \right)^{-1} \\ &= \frac{\Gamma(\alpha_1 + \alpha_2)}{\Gamma(\alpha_1)\Gamma(\alpha_2)} \end{aligned}$$

# Collapsed variational inference [Teh et al. 07]

- If your priors are conjugate, they can be integrated out!
- Let $n_{dzv}$ be the number of times that word v in document d has topic z
- Let $n_{.zv}$ be the number of times word v in any document has topic z

$$p(\mathbf{z}, \mathbf{x} \mid \alpha, \beta) = \prod_d \frac{\Gamma(k\alpha)}{\Gamma\left(k\alpha + n_{d..}\right)} \prod_z \frac{\Gamma\left(\alpha + n_{dz.}\right)}{\Gamma(\alpha)} \prod_z \frac{\Gamma(V\beta)}{\Gamma(V\beta + n_{.z.})} \prod_v \frac{\Gamma(\beta + n_{.zv})}{\Gamma(\beta)}$$

**No theta or phi in the above formula**

# Collapsed variational inference [Teh et al. 07]

- How do we get back phi and theta?

$$\theta_{dz} = \frac{\alpha + n_{dz\cdot}}{K\alpha + n_{d\cdot\cdot}} \qquad \phi_{zv} = \frac{\beta + n_{\cdot zv}}{W\beta + n_{\cdot z\cdot}}$$

- How do we get $n_{dzv}$ ?
  - Gibbs sampling: sample values of Z, count from those examples

# Summary

- Expectation maximization
  - maximizes the lower bound of the likelihood function
  - guarantee to improve the likelihood
  - may stuck at local optimal
- Know how to derive EM algorithm for coin topic and pLSA
- LDA is the Bayesian version of pLSA
  - Direct inference is intractable
  - Variational inference (leveraging ELBO)
  - Markov chain Monte Carlo (Gibbs sampling)

# Resources for LDA

- Mallet: http://mallet.cs.umass.edu/

```
bin/mallet import-dir --input /data/topic-input --output topic-input.mallet \
  --keep-sequence --remove-stopwords
```

- gensim:

```
>>> from gensim.test.utils import common_texts
>>> from gensim.corpora.dictionary import Dictionary
>>>
>>> # Create a corpus from a list of texts
>>> common_dictionary = Dictionary(common_texts)
>>> common_corpus = [common_dictionary.doc2bow(text) for text in common_texts]
>>>
>>> # Train the model on the corpus.
>>> lda = LdaModel(common_corpus, num_topics=10)
```