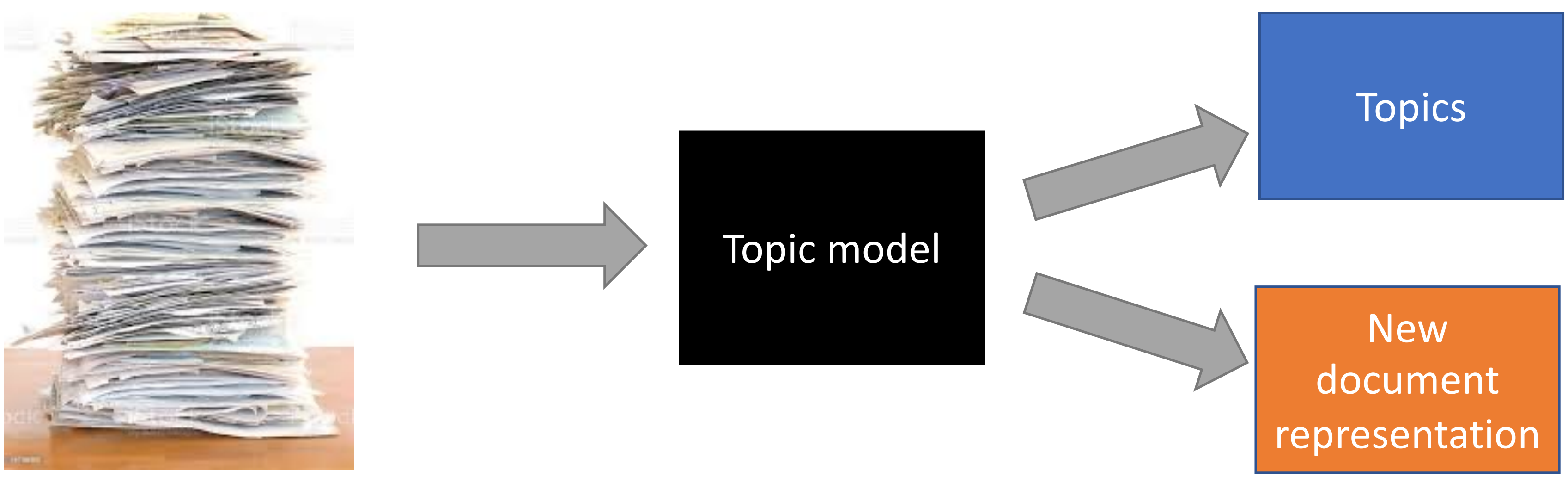
Topic Models

David I. Inouye

__Topic models__ are unsupervised methods for text data that extract topic and document representations

1. Given a dataset of text documents (often called a __**corpus**__), what are the main topics or themes?
2. Can you find a compressed semantic representation of each document/instance?



Motivation: Difficult to discover new and relevant information in uncategorized text collections

- Example: New York Times news articles
  - Automatically categorize articles into different themes
  - How do these themes change over time?
  - What specific articles are in each theme?

- Expensive manual option: Employ many humans to carefully read and categorize

- Cheap automatic option: Use topic models!
  - No labels are required! Just raw text

Other examples that could leverage topic models

- Survey responses
- Customer feedback
- Research papers
- Emails

Overview of topic models

- Motivation
- Preliminary: Representing documents
- Latent Semantic Indexing: Non-probabilistic topic model
  - Mathematical formulation
  - Interpretation of solutions
  - Limitations
- Probabilistic topic models
  - Categorical and multinomial distributions
  - Mixture of multinomials
  - Document-specific mixture of multinomials (LDA)
  - Interpretation
- Algorithms
  - Variational inference (via ELBO as in VAEs)
  - MCMC Gibbs sampling

Preliminary: How should a collection of documents be represented?

- Two naïve assumptions

1. Each word is considered a single unit (called **unigram**)

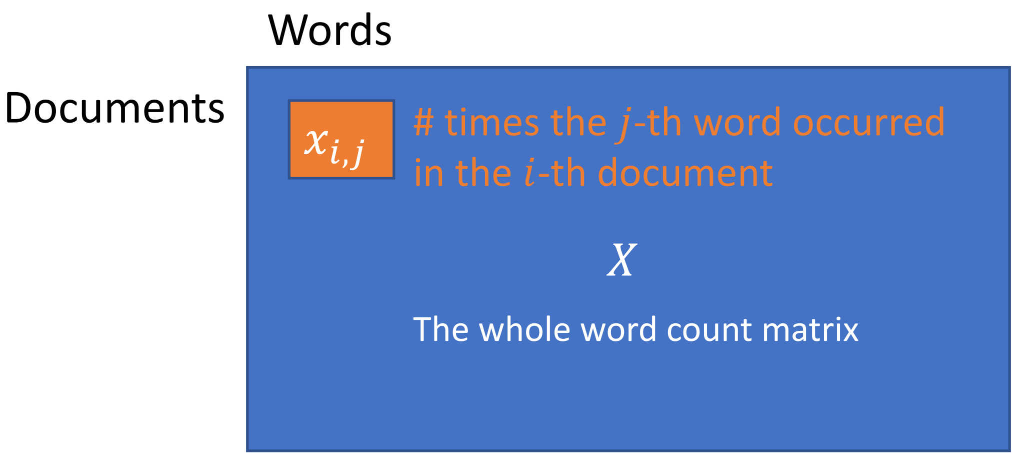
The sun is bright.
The bright sun is red.
----
2 1 3 4
2 4 1 3 5

2. Order of words ignored (**Bag-of-words** assumption)

the sun is bright
=
bright sun the is

Preliminary: The document collection can be represented as a word-count matrix

- Each row represents a document
- Each column represents a word
- Each element represents the number of times (i.e., count) that word occurred in the document



Create word-count matrix in scikit-learn: https://scikit-learn.org/stable/tutorial/text_analytics/working_with_text_data.html

Example word-count matrix

- This movie is very scary and long
- This movie is long and is slow
- This movie is long, spooky good

	1 This	2 movie	3 is	4 very	5 scary	6 and	7 long	8 not	9 slow	10 spooky	11 good
Review 1	1	1	1	1	1	1	1	0	0	0	0
Review 2	1	1	2	0	0	1	1	0	1	0	0
Review 3	1	1	1	0	0	0	1	0	0	1	1

https://www.analyticsvidhya.com/blog/2020/02/quick-introduction-bag-of-words-bow-tf-idf/

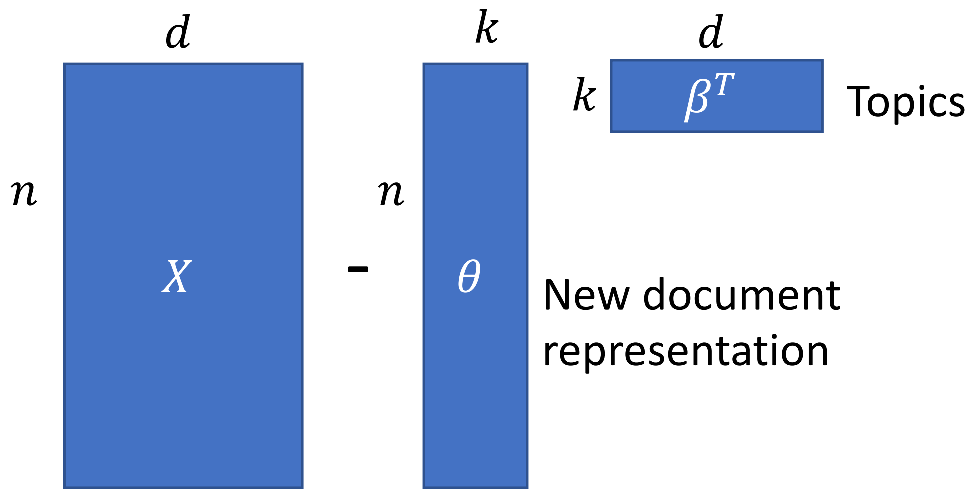
Latent semantic indexing (LSI) is one of the simplest topic models and uses truncated SVD

- Optimization over low rank matrices θ and β

$$\theta, \beta = \min_{\theta,\beta} \|X - \theta\beta^T\|_F^2$$

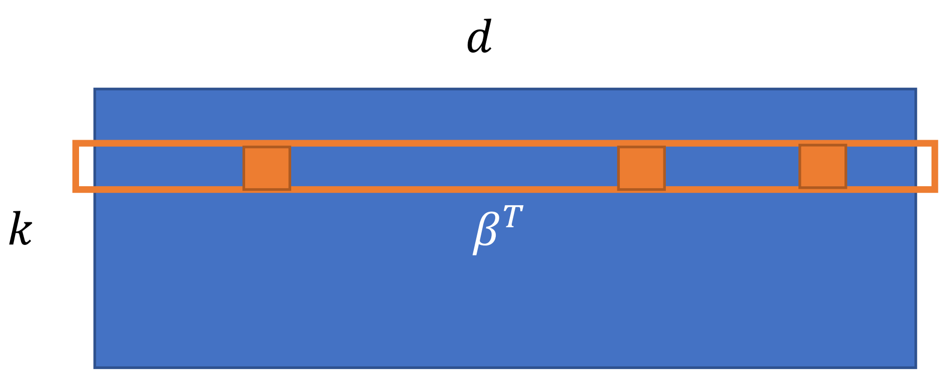
- Solution: Truncated SVD of $X = USV^T$

$$\theta = US_k, \qquad \beta = V_k$$



LSI “topics” can capture <u>synonymy</u> or similarity between words

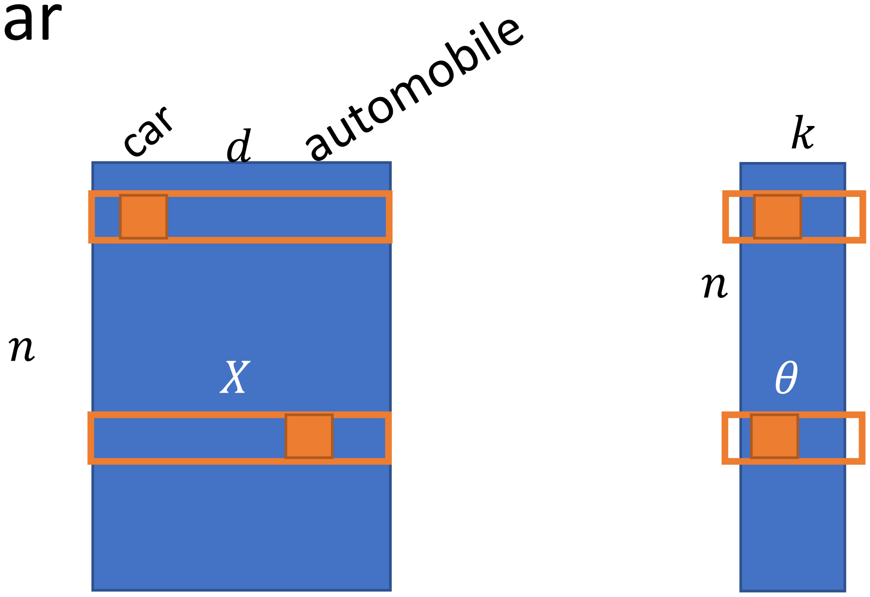
- Examples:
  - “Car” and “automobile” (synonyms)
  - “School” and “education” (related)
- These related words will tend to have high weights in the same row of the topic matrix β^T



“Automotive” topic may have high values on columns for “car”, “automobile” and “truck”.

LSI document representation groups documents even if their exact words do not overlap

- Example
  - One document only uses the word “car”
  - One document only uses the word “automobile”
  - The documents may have no exact words shared but are similar



LSI problem: Interpretation of topics and representations is challenging since values could be arbitrary

- SVD implicitly assume data is real-valued
  - (e.g., -2.1, 3.5, -1.2, 100.1)

- Yet input word-count matrix is discrete data
  - Non-negative integer values (e.g., 0,1,2,3,etc.)

- What do negative values mean?
  (e.g., automobile is 1.1 but school is -0.5)

- What does the scale of these values mean?
  (e.g., 4 or 0.2)

LSI problem: No generative model to create new data (less deep understanding)

- Like the difference between AEs and VAEs
  - VAEs provide a way to generate fake new data

- “What I cannot create, I do not understand.” – Richard Feynman

- Previously we’ve considered mostly *continuous* generative models (GANs, VAEs, flows, etc.)

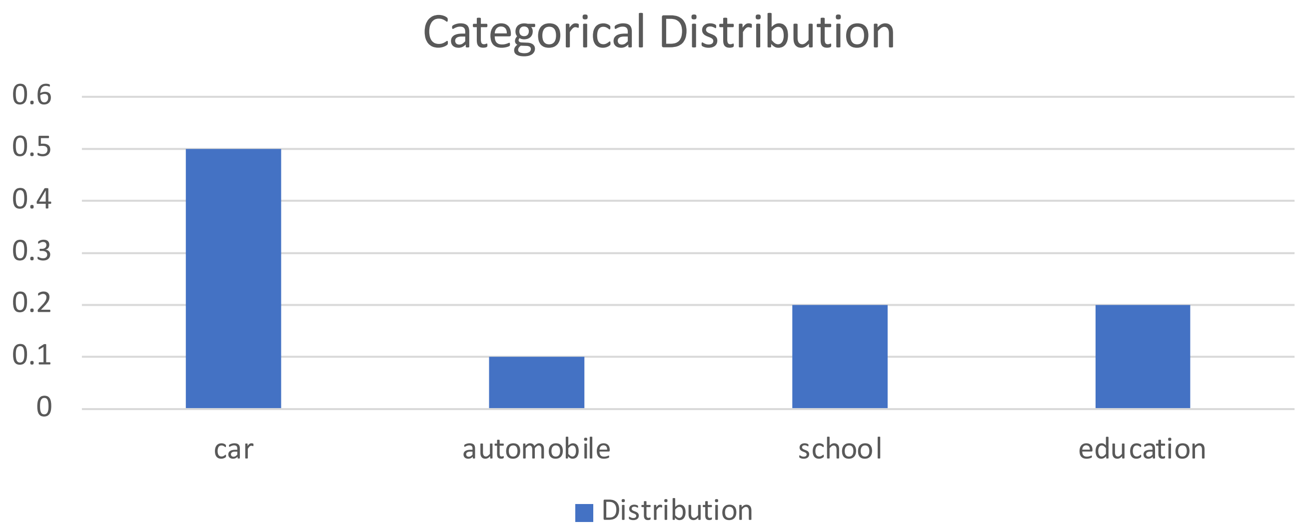
- What about discrete generative models?

A generative model defines the ***assumed*** generative/simulation process of data

- A generative model defines various distributions and how they relate
- The model parameters are not known/given at this stage (more like a template)
  - Learning/training from data comes later
- The assumptions may be very unrealisitic but nonetheless may provide useful information
  - "All models are wrong, some are useful" – George Box
  - Akin to assuming a linear regression model (i.e., probably wrong assumption but still often useful)

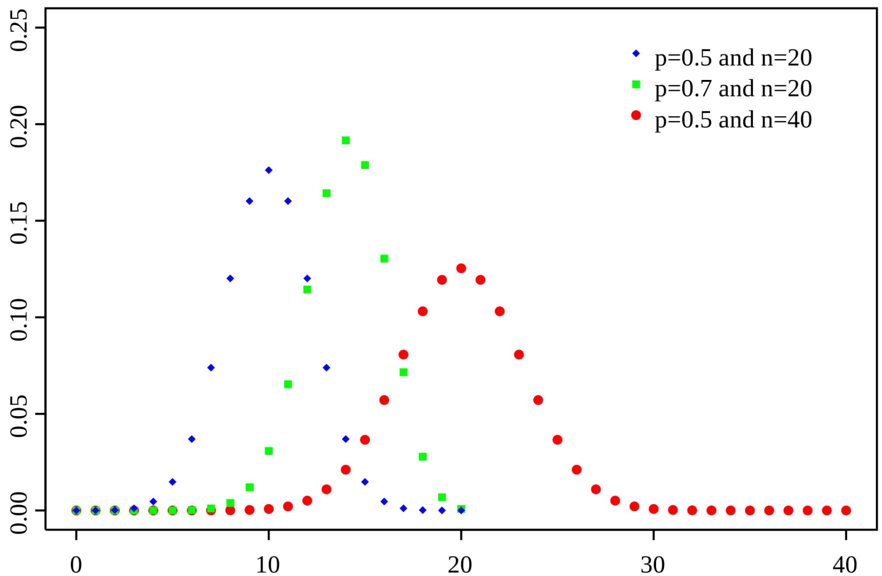
The <u>categorical distribution</u> generalizes the Bernoulli (coin flip) distribution to many outcomes

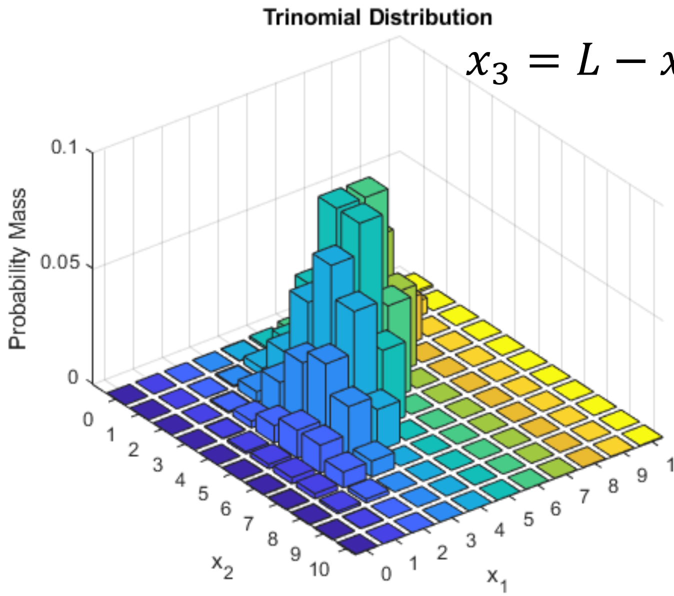
- Intuition, rolling a d-sided dice
- Each side has a probability $p_s = \Pr(x = s)$
- In our case, d is the number of unique words in our corpus



The **multinomial distribution** is a simple model for count data (the “Ind. Gaussian” for count data)

- Intuition, roll d-sided dice L times and record count for each side
- Example: Flip a biased coin 10 times and count how many are heads and tails





$x_3 = L - x_1 - x_2$

The multinomial distribution is a simple model for count data (the "Ind. Gaussian" for count data)

- Word counts can be modeled as
$$x \sim \text{Multinomial}(p; L)$$
  - p is the probability for each word
  - L is the number of words in the document (i.e., length)
    - $L = \sum_s x_s = \|x\|_1$

- Multinomial generative process
  - Repeat $\ell = 1$ to L:
    - Sample individual words $w_{i,\ell} \sim \text{Categorical}(p)$ (where $w_{i,\ell}$ are one hot vectors)
  - $x_i = \sum w_{i,\ell}$ (equivalent to $x_i \sim \text{Multinomial}(p; L)$)

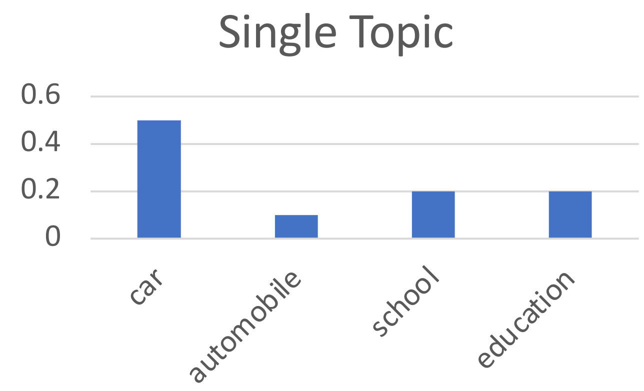
A mixture of multinomials adds complexity like mixture of Gaussians

- Let $x \sim \mathrm{MixtureMult}(\pi, (\beta_1, \cdots, \beta_k); N)$
  - π is the mixture weights
  - β_j is the probability vector for the j-th multinomial component distribution
  - N is the number of words in a document

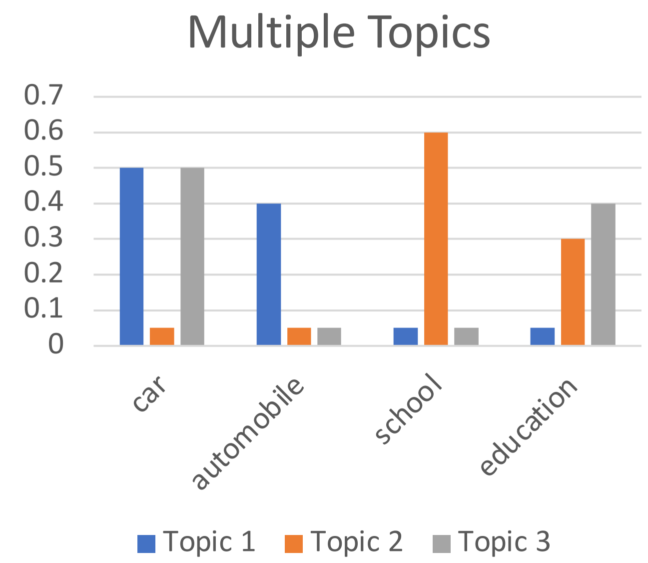
- Mixture generative process (assume L is fixed)
  - Sample single topic $z_i \sim \mathrm{Categorical}(\pi)$
  - Repeat $\ell = 1$ to L:
    - Sample individual words $w_{i,\ell} \sim \mathrm{Categorical}(\beta_{z_i})$ (where w_ℓ are one hot vectors)
  - $x_i = \sum w_{i,\ell}$ (equivalent to $x_i \sim \mathrm{Multinomial}(\beta_{z_i}; L)$)

Interpretation of multinomials and mixture of multinomials

- Multinomial distribution
  - Assumes all documents have the same "topic"
  - A topic is the probability for each word

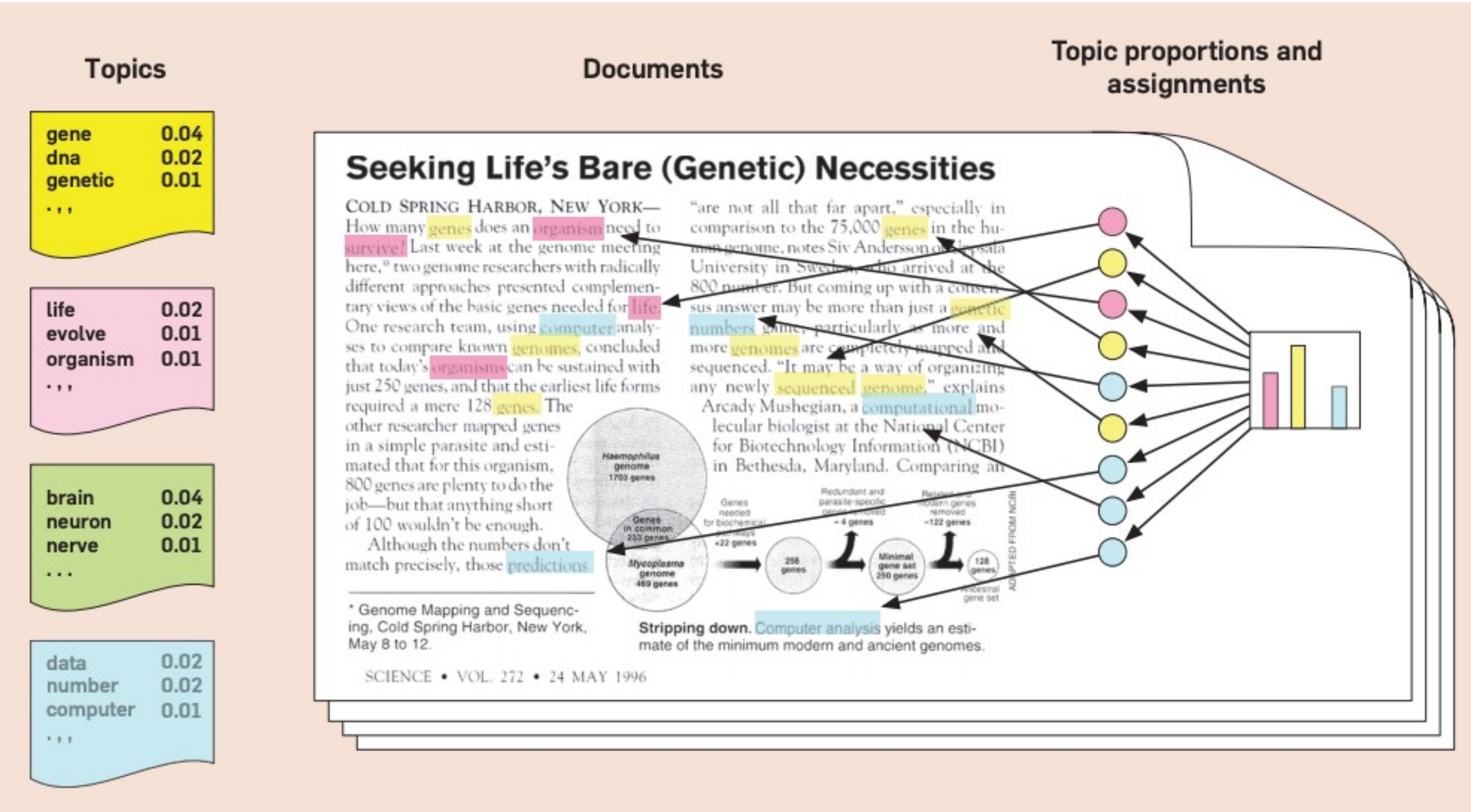


- Multinomial mixture
  - Each component represents a topic
  - Each document only has one topic



- What if each documents have multiple topics?

Document-specific topic mixtures:
<u>Latent Dirichlet Allocation (LDA)</u> defines a model where *each document* can have multiple topics



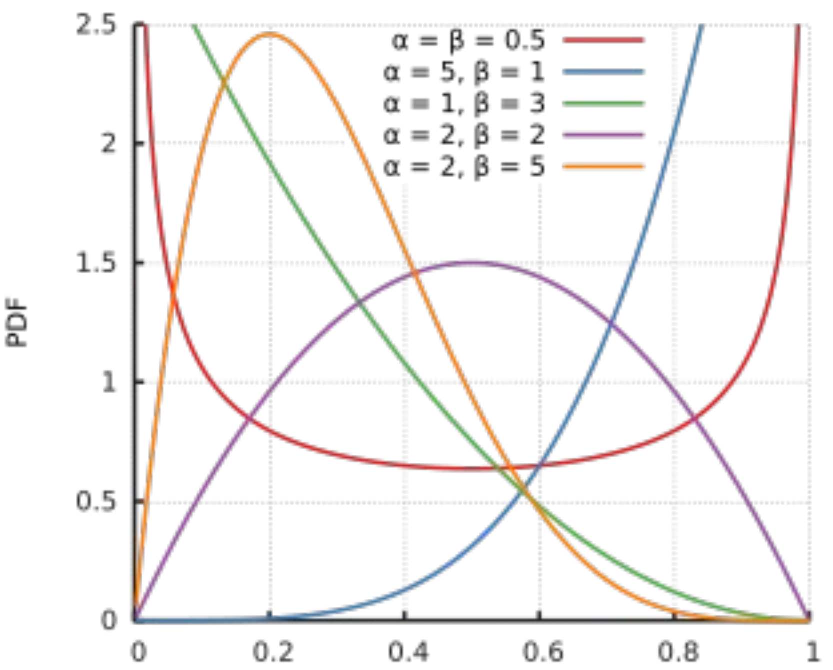
Blei, D. M. (2012). Probabilistic topic models. *Communications of the ACM*, *55*(4), 77-84.

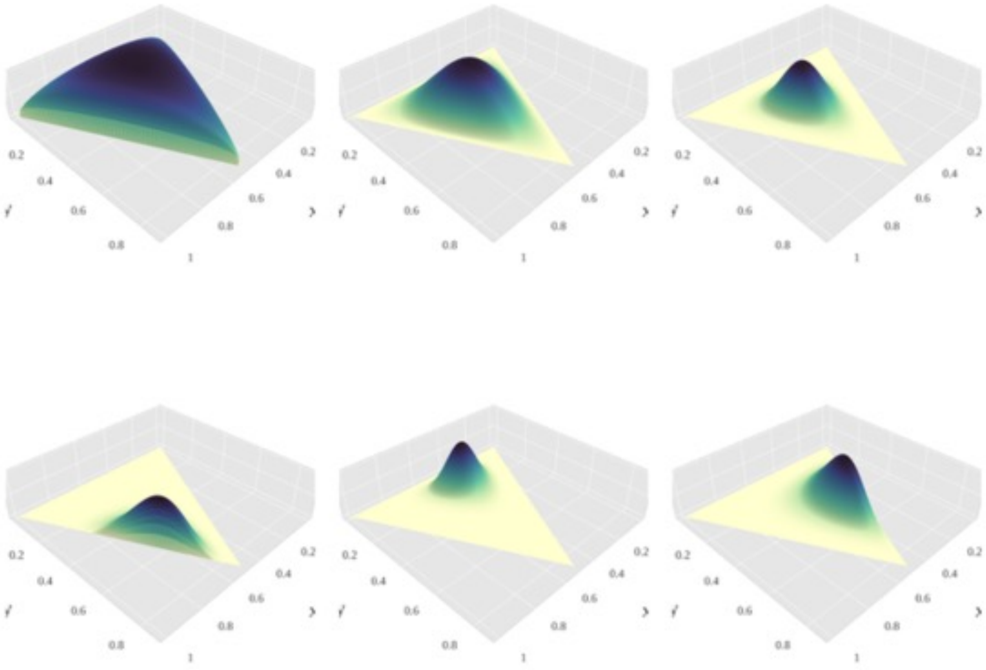
Background: Dirichlet distribution is a distribution over the probability simplex

- The **probability simplex** is the set of vectors that are non-negative and sum to 1

$$\Delta^d := \{x \in [0,1]^d : \sum x_s = 1\}$$

- Dirichlet is simplest distribution on this set

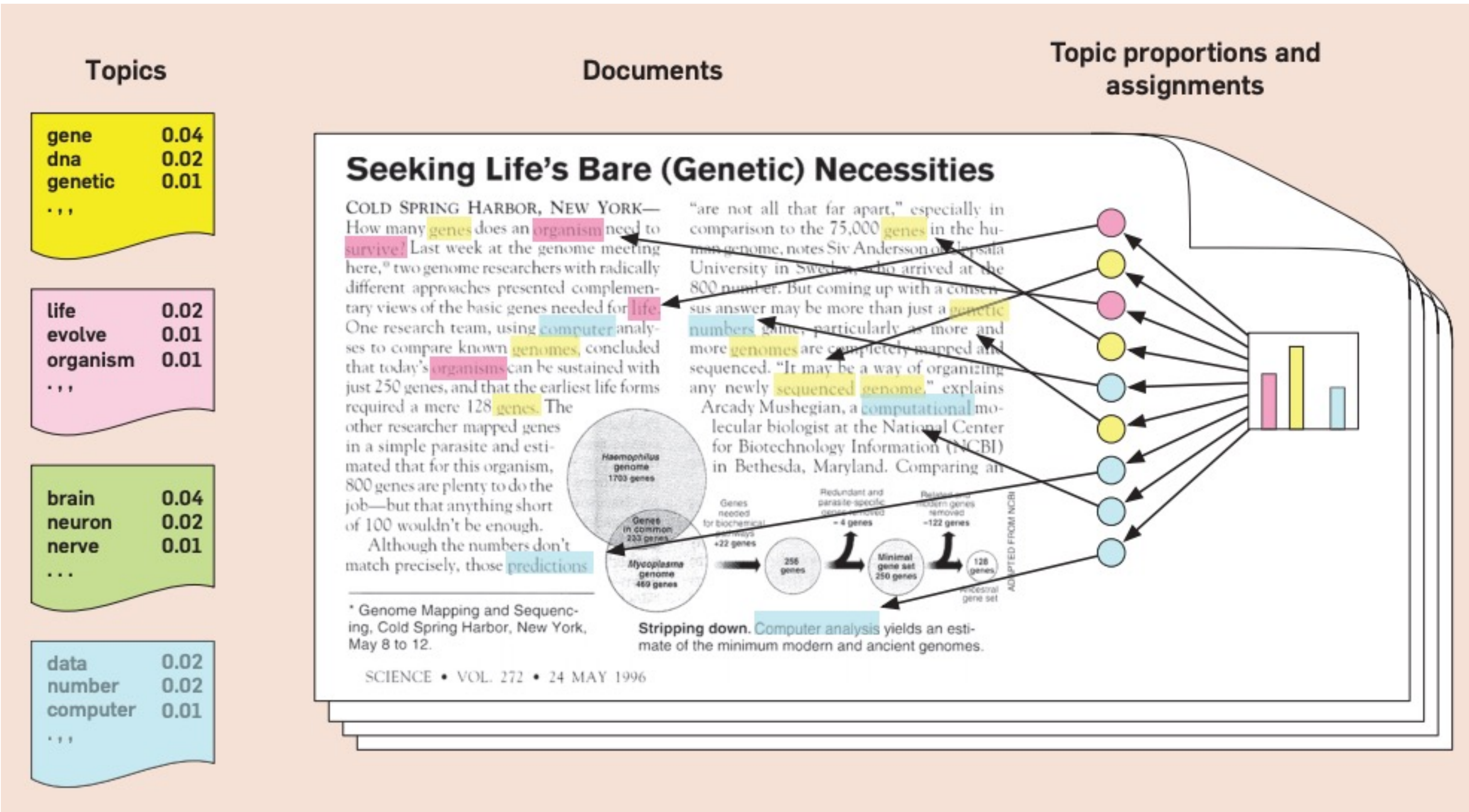




The generative process of LDA is a mixture of mixtures (or admixture)

- Mixture generative process (assume L is fixed)
  - Sample single topic $z_i \sim \text{Categorical}(\pi)$
  - Repeat $\ell = 1$ to L:
    - Sample individual words $w_{i,\ell} \sim \text{Categorical}(\beta_{z_i})$ (where w_ℓ are one hot vectors)
  - $x_i = \sum w_{i,\ell}$ (equivalent to $x_i \sim \text{Multinomial}(\beta_{z_i}; L)$)
- LDA generative process (assume L is fixed)
  - Sample mixture over topics $\theta_i \sim \text{Dirichlet}(\alpha)$
  - Repeat $\ell = 1$ to L:
    - Sample topic of word $z_{i,\ell} \sim \text{Categorical}(\theta_i)$
    - Sample individual words $w_{i,\ell} \sim \text{Categorical}(p_{z_{i,\ell}})$
  - $x_i = \sum w_{i,\ell}$ (equivalent to $x_i \sim \text{Multinomial}(p = \beta\theta_i; L)$)

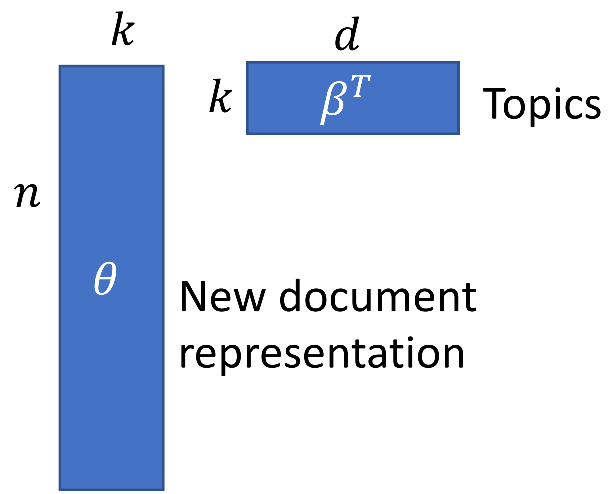
Latent Dirichlet Allocation (LDA) defines a model where each document can have multiple topics



Blei, D. M. (2012). Probabilistic topic models. *Communications of the ACM*, *55*(4), 77-84.

After training, we can recover more interpretable topics and document representations

- Each topic is a probability distribution $\beta_j \in \Delta^d$
- Each document is represented by a probability distribution over topics $\theta_i \in \Delta^k$
- Can be seen as "discrete PCA" method



Estimating these generative models for text data

- Multinomial model
  - MLE has closed form solution (merely empirical frequencies)
- Mixture of multinomials
  - Expectation maximization (EM) algorithm or other mixture-based algorithms
- LDA
  - Variational inference (i.e., ELBO on unknown parameters θ)
  - MCMC/Gibbs sampling (often performs better)

Bayesian inference can be used to **learn/train** model parameters (despite the name)

- **Prior distribution** $p_\alpha(\theta)$ – An **assumed** distribution of the model parameters θ before seeing any data where α is a user-specified hyperparameter.

- **Sampling distribution** $p(X|\theta)$ - The distribution of the training data X given the model parameters θ.

- **Posterior distribution** - The distribution of the parameters after having seen data X.

$$p(\theta|X) = \frac{p(\theta, X)}{p(X)} = \frac{p_\alpha(\theta)p(X|\theta)}{\int p_\alpha(\theta)p(X|\theta)d\theta}$$

- The *mode* or *mean* of the **posterior** $p(\theta|X)$ can provide an estimate for the model parameters θ given training data X.

The *conditional* topic assignment of a single word of LDA given all other topic assignments is known in closed-form

- LDA joint distribution
  (only W is observed, others are latent)

$$p(\theta, \beta, Z, W) \coloneqq p_\eta(\beta) \prod_{i=1}^{n} p_\alpha(\theta_i) \prod_{\ell=1}^{L_i} P(z_{i,\ell} | \theta_i) P(w_{i,\ell} | \beta_{z_{i,\ell}})$$

- **Goal:** Mean or mode of posterior

$$p(\theta, \beta, Z | W) = \frac{p(\theta, \beta, Z, W)}{\int \int \int p(\theta, \beta, Z, W) d\theta d\beta dZ}$$

- **Fact 1:** If Z is known, then obtaining θ and β is easy so we just need Z.

- **Fact 2:** The topic distribution of a *single word* is known in closed-form *conditioned* on the topics of all other words:

$$P(z_{i,\ell} = j | Z_{-(i,\ell)}, W)$$
$$\propto \mathrm{P}(\mathrm{z}_{i,\ell} = \mathrm{j} | Z_{-(i,\ell)}) \mathrm{P}(\mathrm{w}_{i,\ell} | Z_{-(i,\ell)}, W) = \left(\frac{C_{i,j}^{DT} + \alpha}{\sum_{j'} C_{i,j'}^{DT} + \alpha} \right) \left(\frac{C_{w_{i,\ell},j}^{WT} + \eta}{\sum_{w} C_{w,j}^{WT} + \eta} \right)$$

  - $C_{i,j}^{DT}$ is the document-topic counts for document i and topic j
  - $C_{w_{i,\ell},j}^{WT}$ is the word-topic counts for the current word $w_{i,\ell}$ and topic j

Gibbs sampling enables sampling from a joint distribution by only sampling from conditionals

- Gibbs sampling for LDA
  - Randomly initialize Z (like optimization initialization)
  - For $i \in [1,2,\dots,n]$
    - For $\ell \in [1,2,\dots,L_i]$
      - Sample $z_{i,\ell} \sim P\left(z_{i,\ell} = j \middle| Z_{-(i,\ell)}, W\right) \propto= \left(\frac{C_{i,j}^{DT}+\alpha}{\sum_{j'} C_{i,j'}^{DT}+\alpha}\right)\left(\frac{C_{w_{i,\ell},j}^{WT}+\eta}{\sum_{w} C_{w,j}^{WT}+\eta}\right)$
      - (This can be seen as sampling the topic of single word.)
  - Repeat until convergence
- If run long enough, then $Z \sim P(Z|W)$.
- A special type of **Metropolis-Hastings** **Markov Chain Monte Carlo (MCMC)** sampling method

Demo of Gibbs sampling for LDA

Additional resources for topic modeling

- Gentle introduction to topic modeling
  http://www.cs.columbia.edu/~blei/papers/Blei2012.pdf

- More resources/tutorials
  http://www.cs.columbia.edu/~blei/topicmodeling.html

- Nice lecture from CMU on topic models and sampling:
  https://www.cs.cmu.edu/~mgormley/courses/10701-f16/slides/lecture20-topic-models.pdf

- Text analysis with scikit-learn
  https://scikit-learn.org/stable/tutorial/text_analytics/working_with_text_data.html