Master Degree in Computer Science

Natural Language Processing

Topic Modeling

Prof. Alfio Ferrara

Department of Computer Science, Università degli Studi di Milano
Room 7012 via Celoria 18, 20133 Milano, Italia alfio.ferrara@unimi.it

sed noli modo

Motivations for latent semantics discovery

Three problems are typical of Vector Space Models:

High dimensionality of vector representation of documents (that are also sparse).

Synonymy. Many terms have the same meaning but corresponds to different dimensions in the vector space representation. This typically leads to poor recall.

Ambiguity. The same term (same vector dimension) may have different meanings.

Example

Napoleon Bonaparte, later known by his regnal name Napoleon I, was a French military commander and political leader who rose to prominence during the French Revolution and led successful campaigns during the Revolutionary Wars.

Napoleon's political and cultural legacy endures to this day, as a highly celebrated and controversial leader. He initiated many liberal reforms that have persisted in society, and is considered one of the greatest military commanders in history.

Napoleon is a city in Lafayette County, Missouri, and part of the Kansas City metropolitan area within the United States. It is located approximately 30 miles (48 km) east of Kansas City. The population was 222 at the 2010 census.

Napoleon escaped in February 1815 and took control of France. The Allies responded by forming a Seventh Coalition, which defeated Napoleon at the Battle of Waterloo in June 1815. The British exiled him to the remote island of Saint Helena in the Atlantic, where he died in 1821 at the age of 51.

Napoleon lies just a few miles west of Wellington, the two cities having been named after the commanders at the Battle of Waterloo. Approximately halfway between the two cities lies a small, unincorporated crossroads called "Waterloo".

Look at documents as bag-of-(relevant)-words

History

napoleon
bonaparte
napoleon
french
military
commander
political
leader
french
war

History

napoleon
political
leader
military
commander
history

Geography

napoleon
city
county
missouri
united states
kansas
city

History

napoleon
1815
france
napoleon
battle
waterloo
1815
napoleon
1821

History

Geography

napoleon
wellington
city
commander
battle
waterloo
city
waterloo

The computer perspective

From a computer perspective, we **do not know anything about topics** and we **do not know anything about the meaning of words**

The only information is:

- when **two words (strings) are the same**
- when **two words appear in the same document**

```
abc
kfd
abc
chz
mto
crs
pwq
lqw
chz
zxy
```

```
abc
pwq
lqw
mto
crs
hgy
```

```
abc
cyq
ytc
ssi
sus
ksk
cyq
```

```
abc
xxx
chz
abc
bht
woo
xxx
abc
xxy
```

```
abc
wtn
cyq
crs
bht
woo
cyq
woo
```

Our goal is to **aggregate words** in such a way that the **aggregations** (aka **topics**) can **explain the observed documents**

Latent Semantic Indexing

The intuition of LSI (sometimes called Latent Semantic Analysis, LSA) is to discover topics or latent concepts that motivate data even when data use a different terminology for expressing a topic by exploiting matrix factorization techniques.

Quick recap on eigenvalues and eigenvectors

Let *A* be an $n \times n$ matrix with elements being real numbers. If $\overrightarrow{x}$ is an n-dimensional vector, then the matrix-vector product $A\overrightarrow{x}$ is well-defined, and the result is again an n-dimensional vector.

In general, multiplication by a matrix changes the direction of a non-zero vector $\overrightarrow{x}$, unless the vector is special and we have that

$$A\overrightarrow{x} = \lambda\overrightarrow{x}$$

λ is an **eigenvalue**

$\overrightarrow{x}$ is an **eigenvector**

Latent Semantic Indexing

Assuming $A\overrightarrow{x} = \lambda\overrightarrow{x}$, then $(A - \lambda I)\overrightarrow{x} = 0$ where *I* is the identity matrix

Since $\overrightarrow{x}$ is non zero, then $|A - \lambda I| = 0$

In general, for a $n \times n$ matrix, $|A - \lambda I| = 0$ has *n* solutions leading to *n* eigenvalues

According to this result, we can create a matrix using the eigenvalues as follows:

$$AX = A[\overrightarrow{x}_1, \ldots, \overrightarrow{x}_n] = A\overrightarrow{x}_1 + \ldots + A\overrightarrow{x}_n$$

$$= \lambda_1\overrightarrow{x}_1 + \ldots + \lambda_n\overrightarrow{x}_n = [\overrightarrow{x}_1, \ldots, \overrightarrow{x}_n]\begin{bmatrix} \lambda_1 & & \\ & \cdots & \\ & & \lambda_n \end{bmatrix} = [\overrightarrow{x}_1, \ldots, \overrightarrow{x}_n]\Lambda = X\Lambda$$

where Λ is the diagonal matrix of the eigenvalues and $X = [\overrightarrow{x}_1, \ldots, \overrightarrow{x}_n]$

Latent Semantic Indexing

If the *n* eigenvalues are distinct values, then the *n* eigenvectors are linearly independent

Recall that a square matrix *M* is invertible if there exists a square matrix *B* such that $MB = BM = I$. *B* is then the inverse of *M*, denoted M^{-1}

$$AX = X\Lambda \rightarrow AXX^{-1} = X\Lambda X^{-1} \rightarrow A = X\Lambda X^{-1}$$

This representation of *A* is called **diagonalization** of *A*

If *A* is symmetric ($A = A^T$), we have

$$A = X\Lambda X^T$$

Moreover, $n \times n$ symmetric matrices always have real eigenvalues and their eigenvectors are perpendicular

Singular Value Decomposition

Consider a term/document matrix $A_{m \times n}$ with $m > n$. Take $B_{n \times n}$ = $A^T A$.

$$B^T = (A^T A)^T = A^T (A^T)^T = A^T A = B$$

We sort the eigenvalues of *B* and take $\sigma_i^2 = \lambda_i$ for each of the first *k* non-zero ones, so that $V = [\overrightarrow{v}_1, \ldots, \overrightarrow{v}_k]$. We also define $U = [\overrightarrow{u}_1 = \frac{1}{\sigma_1} A \overrightarrow{v}_1, \ldots, \overrightarrow{u}_k = \frac{1}{\sigma_k} A \overrightarrow{v}_k]$ that are perpendicular *m*-dimensional vectors of length 1 ($\overrightarrow{u}_i$ are normalized to 1). We have then:

$$\overrightarrow{u}_j^T A \overrightarrow{v}_i = \overrightarrow{u}_j^T (\sigma_i \overrightarrow{u}_i) = \sigma_i \overrightarrow{u}_j^T \overrightarrow{u}_i$$ That is $$U^T A V = \Sigma \rightarrow A = U \Sigma V^T$$

$\Sigma_{k \times k}$ is the diagonal matrix having $\sigma_1, \ldots, \sigma_k$ along the diagonal. U^T is $k \times m$, *A* is $m \times n$, V is $n \times k$

Singular Value Decomposition

In practice, the matrices *U*, *Σ*, and *V* can be found by transforming *A* in a square matrix and by computing the eigenvectors of this square matrix.

The square matrix is obtained by multiplying the matrix *A* by its transpose. In particular:

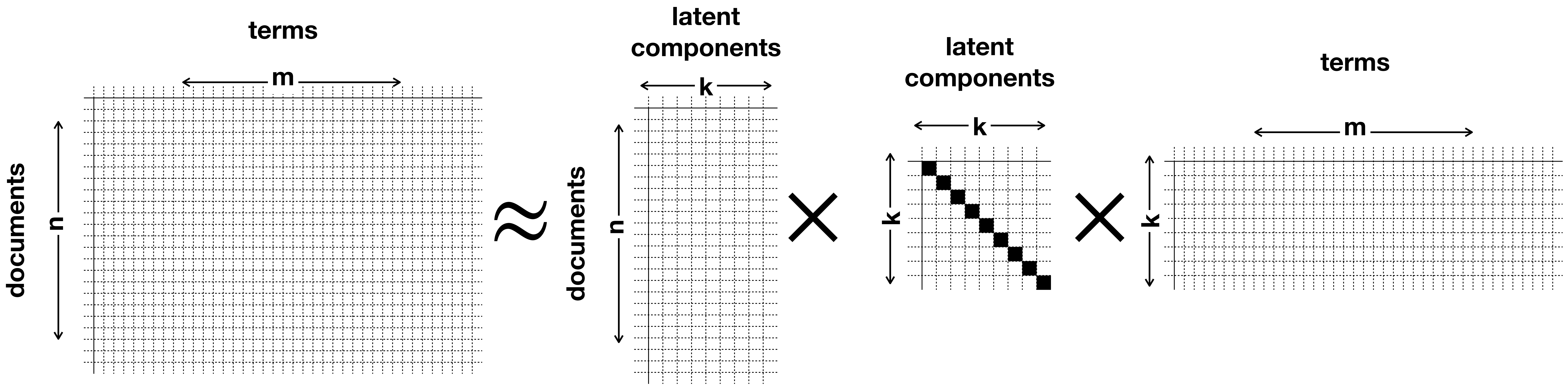
- U is defined by the **eigenvectors** of AA^T
- V is defined by the **eigenvectors** of A^TA
- Σ is the **diagonal matrix composed starting from the eigenvalues** of AA^T and A^TA (which are the same values)

Note that the eigenvalues are involved in two matrix products, that's why we define **the singular values** as $\sigma_i = \sqrt{\lambda_i}$

If the take a number of singular values lower than *n*, SVD becomes a way of reducing the dimensionality of the original matrix.

Singular Value Decomposition

Interpretation of SVD for latent semantic indexing



terms

m

documents

n

$\approx$

latent components

k

documents

n

$\times$

latent components

k

k

$\times$

terms

m

k

A

DOCUMENT-TERM MATRIX

U

TOP-K EIGENVECTORS OF AA^T

Σ

IMPORTANCE OF LATENT COMPONENTS

V^T

TOP-K EIGENVECTORS OF A^TA

UNIVERSITÀ DEGLI STUDI DI MILANO

LA STATALE

Latent Dirichlet Allocation
(aka LDA)

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. Journal of machine Learning research, 3(Jan), 993-1022.

Blei, David, Andrew Ng, and Michael Jordan. "Latent dirichlet allocation." Advances in neural information processing systems 14 (2001).

Introduction to LDA

The main intuition of LDA is to **assume** that the **documents** we observe **are generated by a generative model** defined as follows.

Assume to have K topic distributions that are multinomial distributions over the vocabulary V, where z_i denotes the multinomial distribution of the topic i

Then, we generate each document according to this procedure:

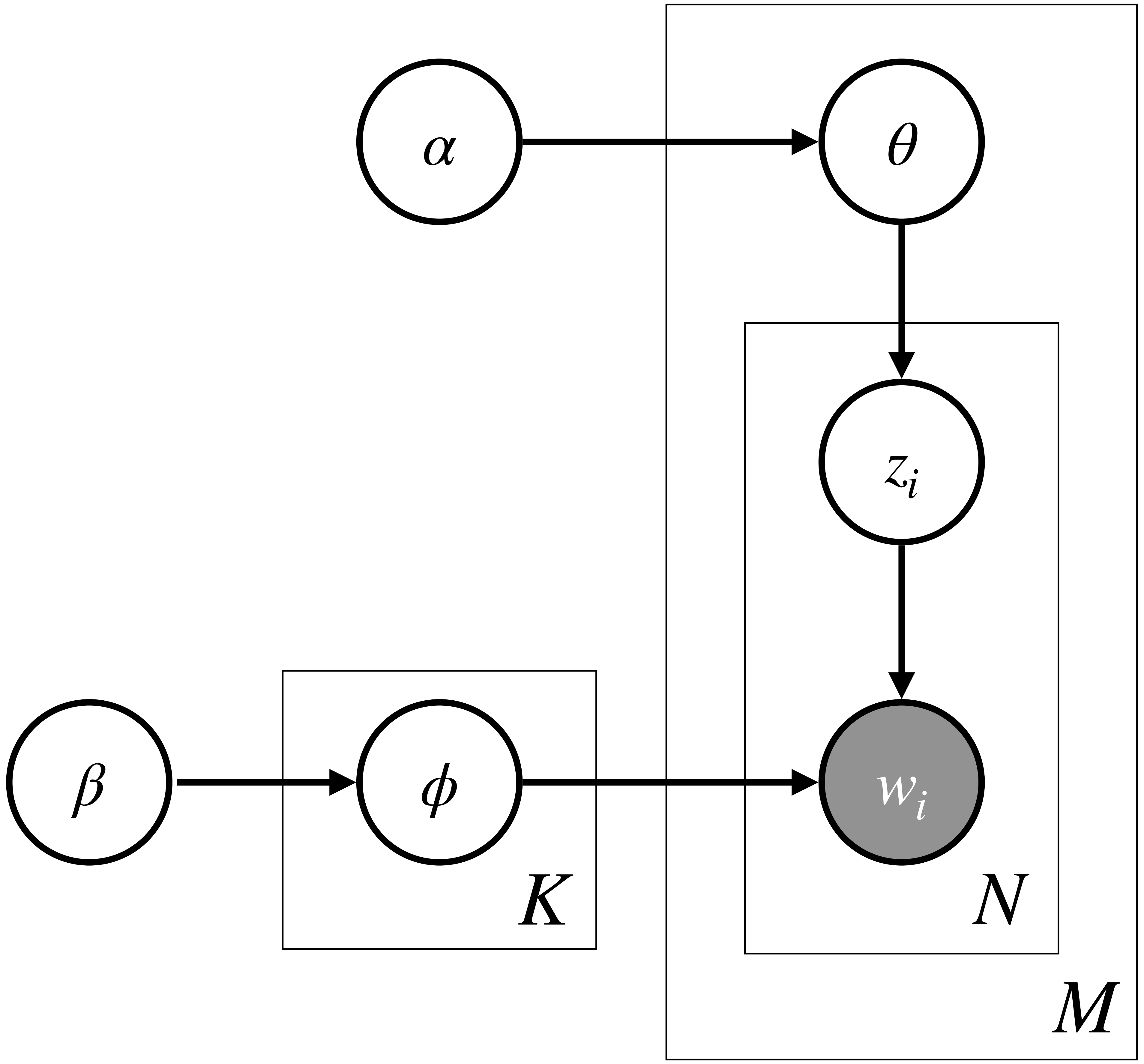
1. Get a multinomial distribution over K
2. Randomly determine the document length N
3. For each word until N
    1. Draw one z_j from the multinomial in (1.)
    2. Draw one word from z_j

Note that. According to this procedure, each **document will be a mixture of multiple topics** (in different proportions) and that each **word is not univocally associated with a single topic**.

Formal definition of LDA

For each document:

1. Choose a topic distribution $\theta_d \sim Dir(\alpha)$, where $Dir(\cdot)$ is drawn from a Dirichlet distribution with parameter α
2. Choose $N \sim Poisson(\zeta)$
3. For each word $w_i : i \in 1 \ldots N$
    1. Choose a topic $z_i \sim Multinomial(\theta_d)$
    2. Select $\phi^{(z_i)} \sim Dir(\beta)$
    3. Choose a word w_i from $P(w_i \mid z_i, \beta) \sim \phi^{(z_i)}$, a multinomial probability conditioned on the topic z_i

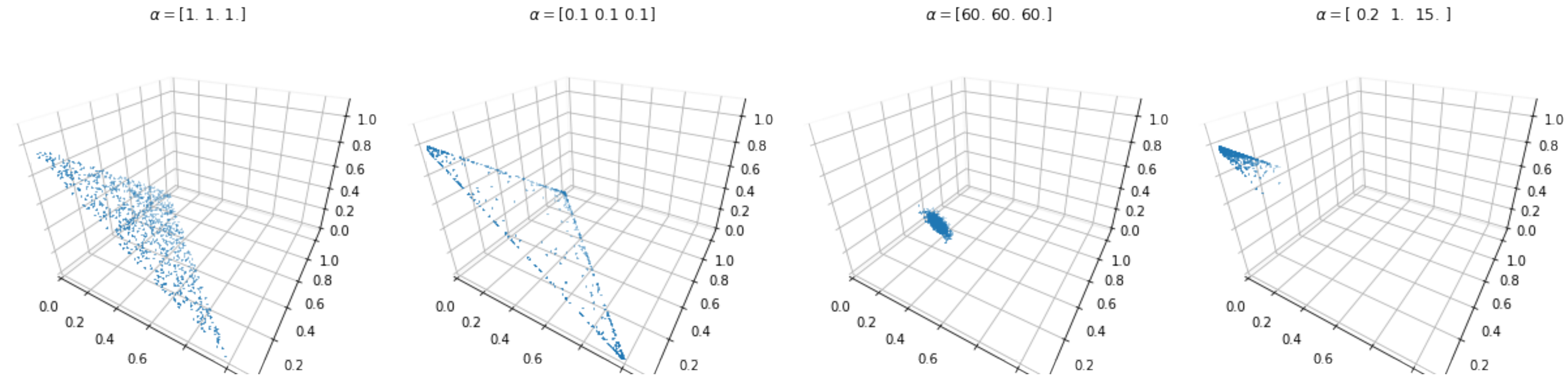


Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. Journal of machine Learning research, 3(Jan), 993-1022.

Dirichlet distribution

UW, U. (2010). Introduction to the Dirichlet distribution and related processes. https://vannevar.ece.uw.edu/techsite/papers/documents/UWEETR-2010-0006.pdf

One application area where the Dirichlet has proved to be particularly useful is in modeling the distribution of words in text documents. If we have a **dictionary containing k possible words**, then a particular document can be represented by a **probability mass function (pmf)** of length k produced by **normalizing the empirical frequency of its words**. A group of documents produces a collection of pmfs, and we can fit a Dirichlet distribution to capture the variability of these pmfs.



A draw from a k dimensional Dirichlet distribution returns a k dimensional multinomial, θ in this case, where the k values must sum to one. θ has the probability density:

$$P(\theta \mid \alpha) = \frac{\Gamma\left(\sum_{i=1}^{k} \alpha_i\right)}{\prod_{i=1}^{k} \Gamma(\alpha_i)} \prod_{i=1}^{k} \theta_i^{\alpha_i - 1}$$

$$\Gamma(\alpha) = \int_0^\infty x^{\alpha-1} e^{-x}\, dx, \textbf{ with } \alpha > 0$$

Probability of a document under the generative model

For each document

For each topic

For each word

$$P(W, z, \theta, \phi, \alpha, \beta) = \prod_{j=1}^{M} P(\theta_j; \alpha) \prod_{i=1}^{K} P(\phi_i; \beta) \prod_{t=1}^{N} P(z_{j,t} \mid \theta_j) \; P(W_{j,t} \mid \phi^{(z_{j,t})})$$

Dirichlet → Pick up **topic**

Dirichlet → Pick up **words**

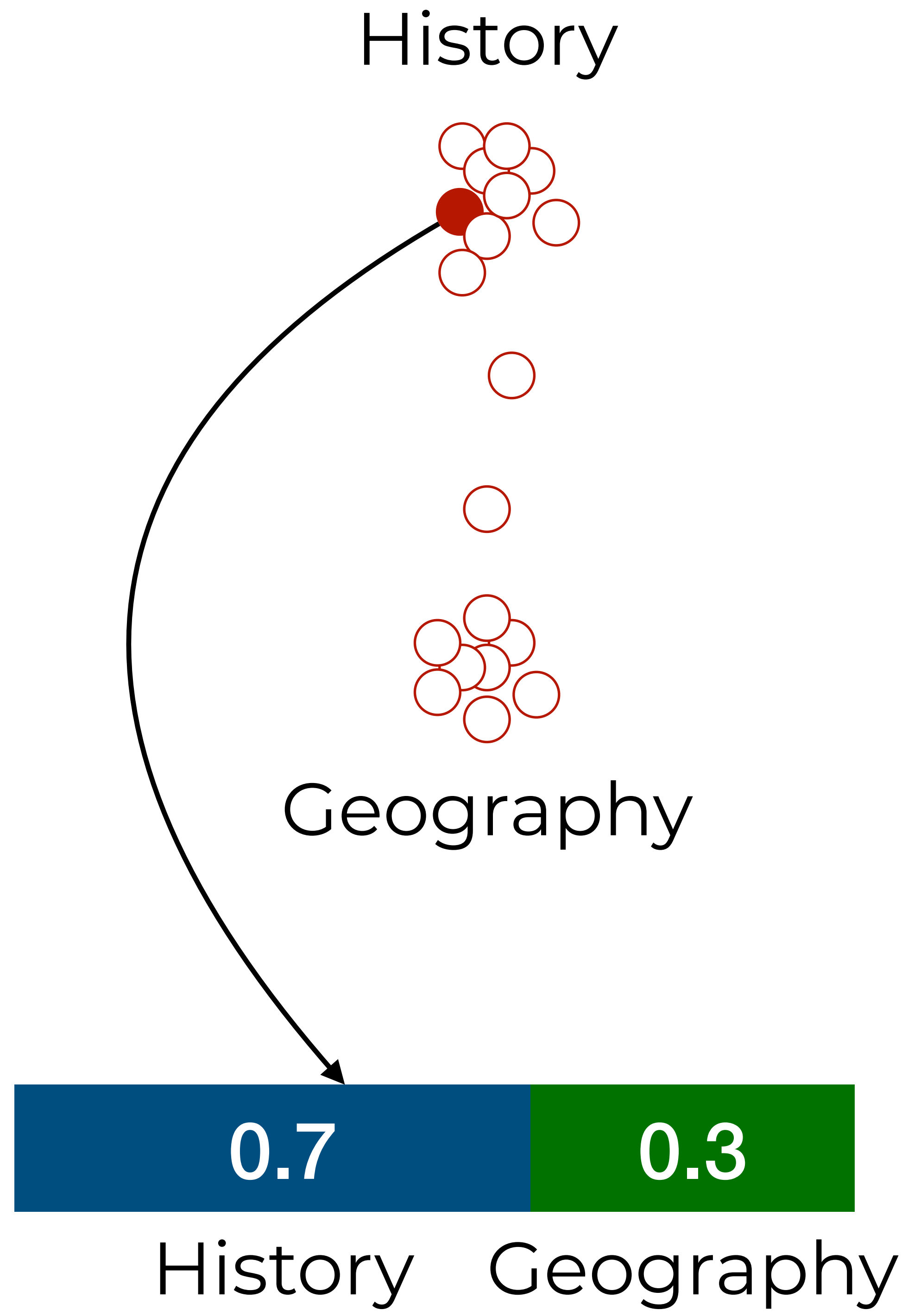
Multinomial → Pick up **topic**

Multinomial → Pick up **words**

Probability of a document under the generative model

$$\prod_{j=1}^{M} P(\theta_j;\alpha) \quad \prod_{i=1}^{K} P(\phi_i;\beta) \quad \prod_{t=1}^{N} \quad P(z_{j,t} \mid \theta_j) \quad P(W_{j,t} \mid \phi^{(z_{j,t})})$$

Draw a distribution



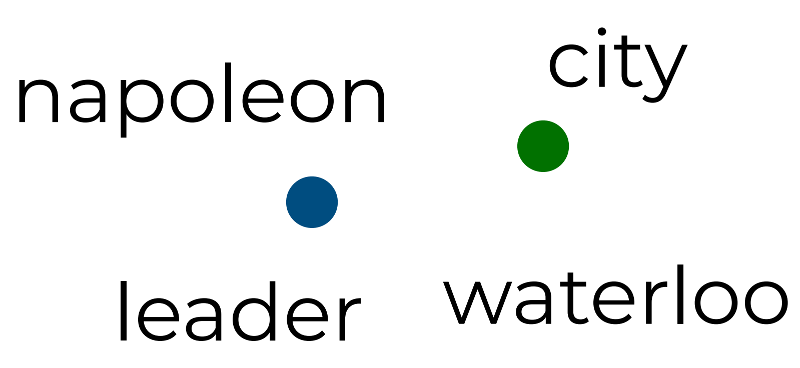
History

Geography

0.7 History

0.3 Geography

Draw a distribution



napoleon

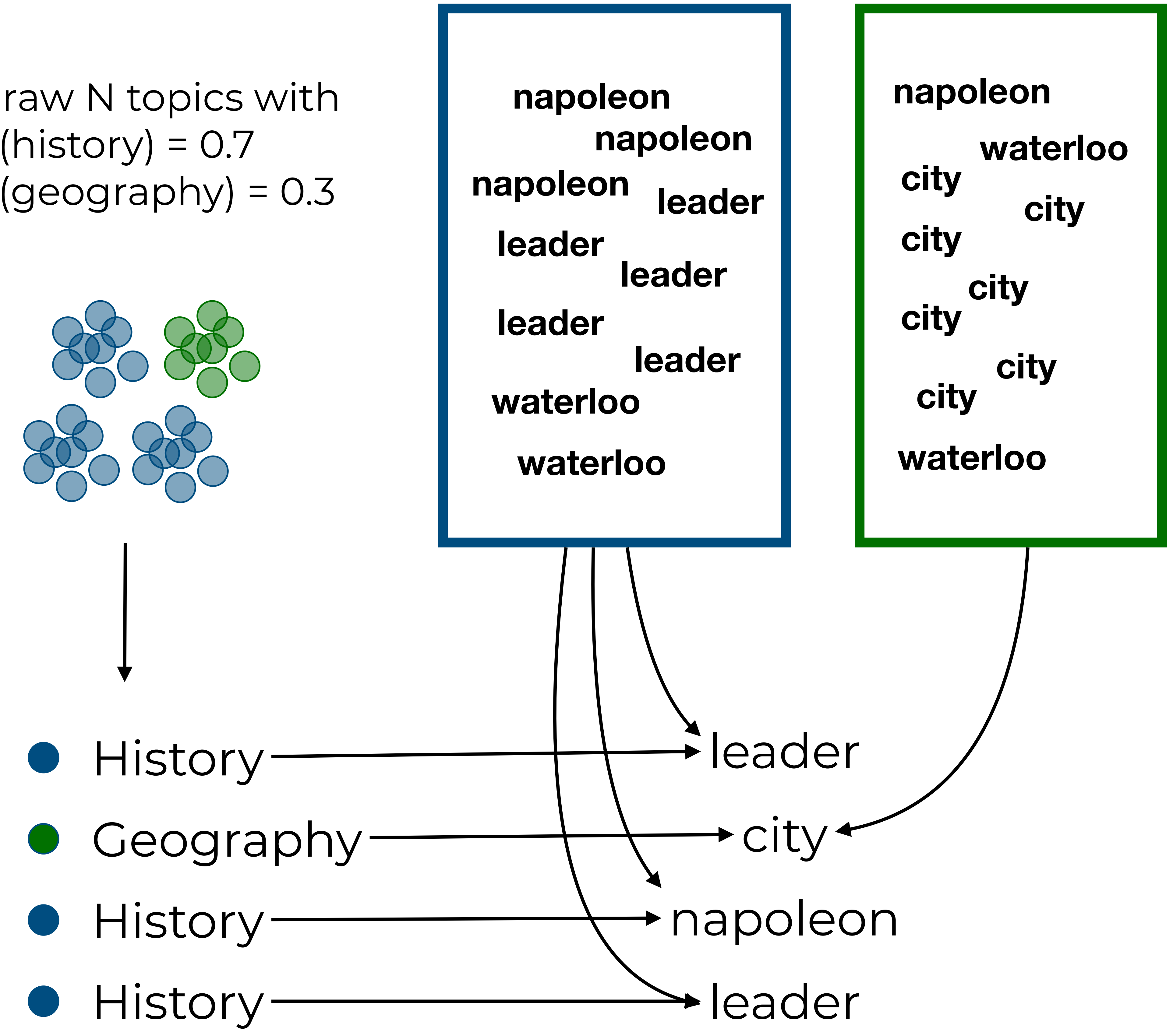
city

leader

waterloo

Topic	napoleon	leader	waterloo	city
history	0.3	0.5	0.2	0.0
geography	0.1	0.0	0.2	0.7

Draw N topics with
P(history) = 0.7
P(geography) = 0.3



napoleon
napoleon
napoleon
leader
leader
leader
leader
leader
waterloo
waterloo

napoleon
waterloo
city
city
city
city
city
city
city
waterloo

● History → leader

● Geography → city

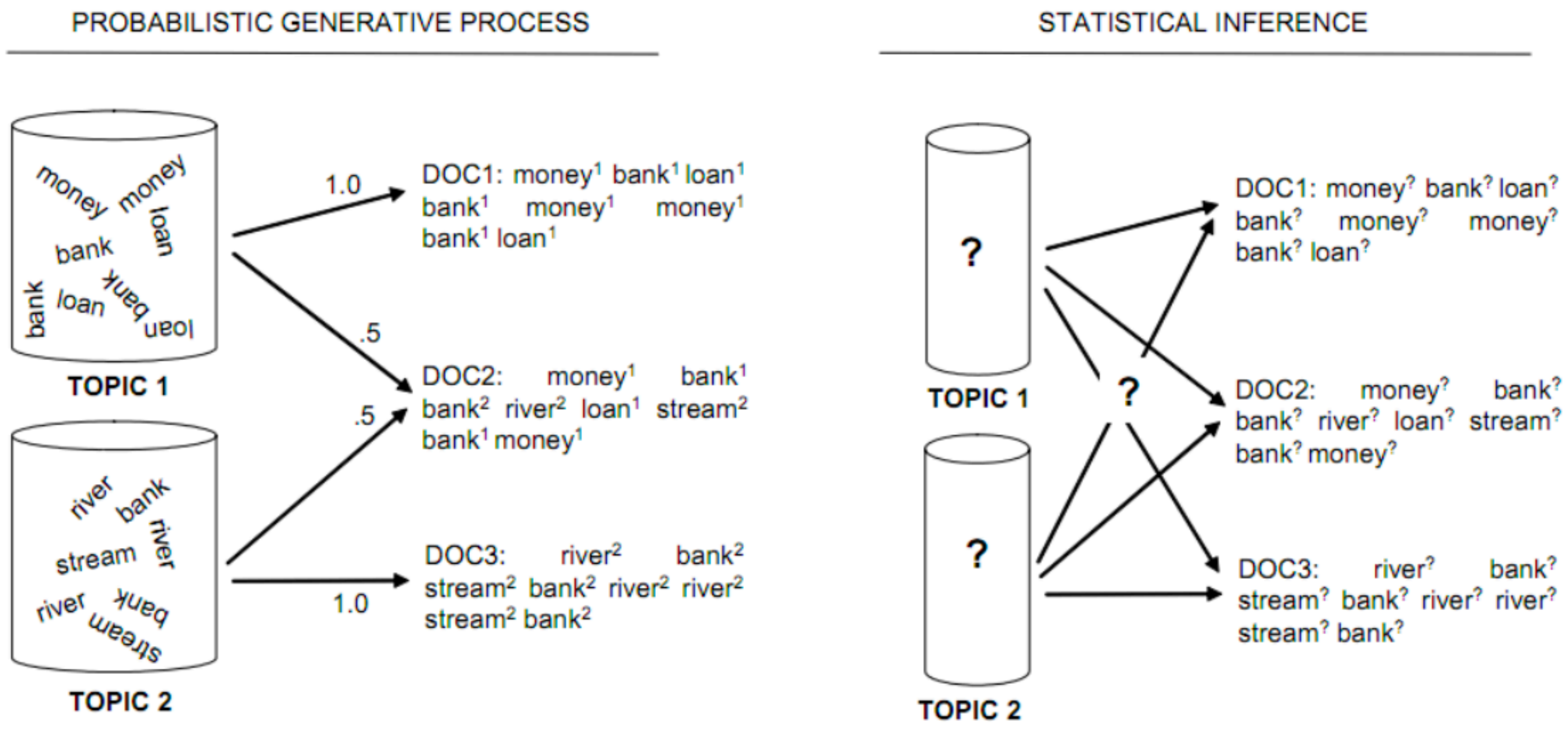
● History → napoleon

● History → leader

Example inspired to https://www.youtube.com/watch?v=T05t-SqKArY

Problem

Statistical inference. In a real world scenario we do not have the distributions θ and ϕ nor α and β. Moreover, we do not aim at generating documents but instead at detecting the topics out of a corpus of text documents.
Thus, we need to **learn the model from the corpus (*posterior inference*)**.
There are several techniques, including **variational inference** (as used in the original LDA paper) and **Gibbs Sampling** (as we will use here).


PROBABILISTIC GENERATIVE PROCESS
money money loan bank bank loan bank loan
TOPIC 1
river bank river stream river bank stream
TOPIC 2
1.0
.5
.5
1.0
DOC1: money^1 bank^1 loan^1 bank^1 money^1 money^1 bank^1 loan^1
DOC2: money^1 bank^1 bank^2 river^2 loan^1 stream^2 bank^1 money^1
DOC3: river^2 bank^2 stream^2 bank^2 river^2 river^2 stream^2 bank^2
STATISTICAL INFERENCE
?
TOPIC 1
?
?
TOPIC 2
DOC1: money$^?$ bank$^?$ loan$^?$ bank$^?$ money$^?$ money$^?$ bank$^?$ loan$^?$
DOC2: money$^?$ bank$^?$ bank$^?$ river$^?$ loan$^?$ stream$^?$ bank$^?$ money$^?$
DOC3: river$^?$ bank$^?$ stream$^?$ bank$^?$ river$^?$ river$^?$ stream$^?$ bank$^?$

M. Steyvers and T. Griffiths. Probabilistic topic models. Handbook of latent semantic analysis, 427(7):424–440, 2007.

UNIVERSITÀ
DEGLI STUDI
DI MILANO

LA STATALE

Gibbs sampling for LDA statistical inference

Darling, W. M. (2011, December). A theoretical and practical implementation tutorial on topic modeling and gibbs sampling. In Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies (pp. 642-647).

Posterior inference

Posterior inference means to reversing the defined generative process and learning the posterior distributions of the latent variables in the model given the observed data, in order to **maximize the probability of the model to generate the given corpus**. This is modeled by the following intractable equation

$$P(\theta, \phi, \mathbf{z} \mid \mathbf{w}, \alpha, \beta) = \frac{P(\theta, \phi, \mathbf{z}, \mathbf{w} \mid \alpha, \beta)}{P(\mathbf{w} \mid \alpha, \beta)}$$

Gibbs sampling is one member of a family of algorithms from the Markov Chain Monte Carlo (MCMC) framework. The MCMC algorithms aim to construct a Markov chain such that, after a number of iterations of stepping through the chain, sampling from the distribution should converge to be close to sampling from the desired posterior .

The idea of Gibbs sampling

Suppose we want to sample x from the joint distribution $P(\mathbf{x}) = P(x_1, \ldots, x_n)$

1. At time $t = 0$, randomly initialize $\mathbf{x}$
2. For $t \in 1 \ldots T$:
    1. $x_1^t \sim P(x_1 \mid x_2^{(t-1)}, x_3^{(t-1)}, \ldots, x_n^{(t-1)}$
    2. $x_2^t \sim P(x_2 \mid x_1^{(t-1)}, x_3^{(t-1)}, \ldots, x_n^{(t-1)}$
    3. …
    4. $x_n^t \sim P(x_n \mid x_2^{(t-1)}, x_3^{(t-1)}, \ldots, x_{n-1}^{(t-1)}$

At each iteration, we assume that all the other values except the one we are observing are correct and we use them to update the value at hand. This is repeated for all the values and for several iterations.

In LDA we need to estimate three latent variables, that are θ_d (document-topic distribution), $\phi^{(z)}$ (topic-word distribution), and z_i (topic assignment to words). However, $\mathbf{z}$ is sufficient to derive both θ_d and $\phi^{(z)}$. The idea is then to estimate just $\mathbf{z}$. This is called **collapsed Gibbs sampling**.

$$\theta_{d;z} = \frac{count(d, z) + \alpha}{\sum_{i=1}^{K} count(d, z_i) + \alpha}$$

$$\phi_{z;w} = \frac{count(z, w) + \beta}{\sum_{i=1}^{V} count(z, w_i) + \beta}$$

Collapsed Gibbs sampling

The collapsed Gibbs sampler for LDA needs to compute the probability of a topic z being assigned to a word w_i, given all other topic assignments to all other words

$$P(z_i \mid \mathbf{z}_{\neg i}, \alpha, \beta, \mathbf{w})$$

This can be rewritten as

$$P(z_i \mid \mathbf{z}_{\neg i}, \alpha, \beta, \mathbf{w}) = \frac{P(z_i, \mathbf{z}_{\neg i}, \mathbf{w} \mid \alpha, \beta)}{P(\mathbf{z}_{\neg i}, \mathbf{w} \mid \alpha, \beta)} \approx P(z_i, \mathbf{z}_{\neg i}, \mathbf{w} \mid \alpha, \beta) = P(\mathbf{z}, \mathbf{w} \mid \alpha, \beta)$$

Which depends on ϕ and θ as follows

$$P(\mathbf{z}, \mathbf{w} \mid \alpha, \beta) = \int\int P(\mathbf{z}, \mathbf{w}, \theta, \phi \mid \alpha, \beta)\ d\theta\ d\phi$$

Collapsed Gibbs sampling

$$P(\mathbf{z}, \mathbf{w} \mid \alpha, \beta) = \int\int P(\mathbf{z}, \mathbf{w}, \theta, \phi \mid \alpha, \beta)\ d\theta\ d\phi$$

↓

LDA definition

↓

$$P(\mathbf{z}, \mathbf{w} \mid \alpha, \beta) = \int\int P(\phi \mid \beta) P(\theta \mid \alpha) P(\mathbf{z} \mid \theta) P(\mathbf{w} \mid \phi^{(z)})\ d\theta\ d\phi$$

On which we can group dependent variables

$$= \int P(\mathbf{z} \mid \theta) P(\theta \mid \alpha)\ d\theta\ \int P(\mathbf{w} \mid \phi^{(z)}) P(\phi \mid \beta)\ d\phi$$

This way, we obtain two terms that are both multinomial distributions with Dirichlet priors

Collapsed Gibbs sampling

We can use the Dirichlet definition and the $B(\alpha)$ multinomial function, that is $B(\alpha) = \dfrac{\prod_k \Gamma(\alpha_k)}{\Gamma\left(\sum_k \alpha_k\right)}$

Documents-topics

$$\int P(\mathbf{z} \mid \theta)P(\theta \mid \alpha)\ d\theta$$

$$= \int \prod_i \theta_{d,z_i} \frac{1}{B(\alpha)} \prod_k \theta_{d,k}^{\alpha_k}\ d\theta_d$$

$$= \frac{1}{B(\alpha)} \int \prod_k \theta_{d,k}^{count(d,k)+\alpha_k}\ d\theta_d$$

$$= \frac{B(count(d,\cdot\,)+\alpha)}{B(\alpha)}$$

Topic-words

$$\int P(\mathbf{w} \mid \phi^{(z)})P(\phi \mid \beta)\ d\phi$$

$$= \int \prod_d \prod_i \phi_{z_{d,i},w_{d,i}} \prod_k \frac{1}{B(\beta)} \prod_w \phi_{k,w}^{\beta_w}\ d\phi_k$$

$$= \prod_k \frac{1}{B(\beta)} \int \prod_w \phi_{k,w}^{\beta_w+count(k,w)}\ d\phi_k$$

$$= \prod_k \frac{B(count(k,\cdot\,)+\beta)}{B(\beta)}$$

Collapsed Gibbs sampling

Combining the previous equations, we finally obtain

$$P(\mathbf{z}, \mathbf{w} \mid \alpha, \beta) = \prod_d \frac{B(count(d, \cdot\,) + \alpha)}{B(\alpha)} \quad \prod_k \frac{B(count(k, \cdot\,) + \beta)}{B(\beta)}$$

And the update rule for Gibbs sampling (parameters are omitted)

$$P(z_i \mid \mathbf{z}^{\neg i}, \mathbf{w}) = \frac{P(\mathbf{w}, \mathbf{z})}{P(\mathbf{w}, \mathbf{z}^{\neg i})} \quad = \frac{P(\mathbf{z})}{P(\mathbf{z}^{\neg i})} \cdot \frac{P(\mathbf{w} \mid \mathbf{z})}{P(\mathbf{w}^{\neg i} \mid \mathbf{z}^{\neg i})}$$

$$\propto \prod_d \frac{B(count(d, \cdot\,) + \alpha)}{B(count(d, \cdot\,)^{\neg i} + \alpha)} \prod_k \frac{B(count(k, \cdot\,) + \beta)}{B(count(k, \cdot\,)^{\neg i} + \beta)}$$

Collapsed Gibbs sampling (implementation)

Input: words $\mathbf{w}$ and documents $\mathbf{d}$

Output: topic assignments $\mathbf{z}$ and $count(d, k)$, $count(k, w)$, $count(k)$

Let **N** be the number of words in the corpus

```
Begin
    - randomly initialize $\mathbf{z}$ and update counters
    - for iteration in 1 ... T:
        - for $i = 0 \rightarrow N - 1$ do:
            - $word = w[i],\ topic = z[i]$
            - reduce by 1 the counters for word and its corresponding document
            - for $k = 0 \rightarrow K - 1$ do:
                - $P(z = k \mid \cdot\,) = (count(d, k) + \alpha_k)\frac{count(k, w) + \beta_w}{count(k) + \beta \cdot \mid V \mid}$
            - $new\_topic$ = sample from $P(z \mid \cdot\,)$
            - $z[i] = new\_topic$
            - update counters
End
```