Information Retrieval & Data Mining [COMP0084]

Topic models and vector semantics

Vasileios Lampos

Computer Science, UCL

lampos.net

Preliminaries

- In these lectures:
  - Introduction to topic models
  - Introduction to vector semantics

- Useful additional material
  - "*Speech and language processing*" (Jurafsky, Martin), web.stanford.edu/~jurafsky/slp3/
  - pLSA (Hofmann), iro.umontreal.ca/~nie/IFT6255/Hofmann-UAI99.pdf
  - LDA (Blei, Ng, Jordan), jmlr.org/papers/volume3/blei03a/blei03a.pdf
  - word2vec (Mikolov et al.), arxiv.org/abs/1301.3781
  - Blei on LDA, youtube.com/watch?v=DDq3OVp9dNA
  - Boyd-Graber on topic models, youtube.com/watch?v=yK7nN3FcgUs
  - Manning on word2vec, youtube.com/watch?v=ERibwqs9p38

- Some slides adapted from WSDM '14 tutorial on "Multilingual Probabilistic Topic Modelling" — liir.cs.kuleuven.be/tutorial/WSDM2014Tutorial.pdf

What is a topic model?

What is a topic model?

- **Informally:** groupings (or clusters) of words (terms, *n*-grams) that are somehow related

What is a topic model?

- **Informally:** groupings (or clusters) of words (terms, *n*-grams) that are somehow related
- **Still informally:** method for automatically organising, understanding, searching, and summarising large (digitised) document collections
  — uncovers hidden (*latent*) topical patterns (**topics!**) in the collection
  — can annotate, and then organise or summarise, the documents based on these topics

What is a topic model?

- **Informally:** groupings (or clusters) of words (terms, *n*-grams) that are somehow related
- **Still informally:** method for automatically organising, understanding, searching, and summarising large (digitised) document collections
  — uncovers hidden (*latent*) topical patterns (**topics!**) in the collection
  — can annotate, and then organise or summarise, the documents based on these topics
- As we will see, it is often defined as a **probabilistic structure** expressing a certain set of assumptions about how the documents in our collection were generated

What is a topic model?

- **Informally:** groupings (or clusters) of words (terms, *n*-grams) that are somehow related
- **Still informally:** method for automatically organising, understanding, searching, and summarising large (digitised) document collections
  — uncovers hidden (*latent*) topical patterns (**topics!**) in the collection
  — can annotate, and then organise or summarise, the documents based on these topics
- As we will see, it is often defined as a **probabilistic structure** expressing a certain set of assumptions about how the documents in our collection were generated
- **Note:** we can also learn topic models (word clusters) using clustering techniques with no explicit probabilistic structure such as k-means

Why do need topics?

Why do need topics?

- Too **many documents** and we can’t read them all!

Why do need topics?

- Too **many documents** and we can’t read them all!
- Topic models can automatically **categorise** large document collections, so that we can browse through them much more efficiently
- Topic models can be applied on **various corpus collections**, attracting inter-disciplinary interest
  e.g. newspapers, books, social media, health reports

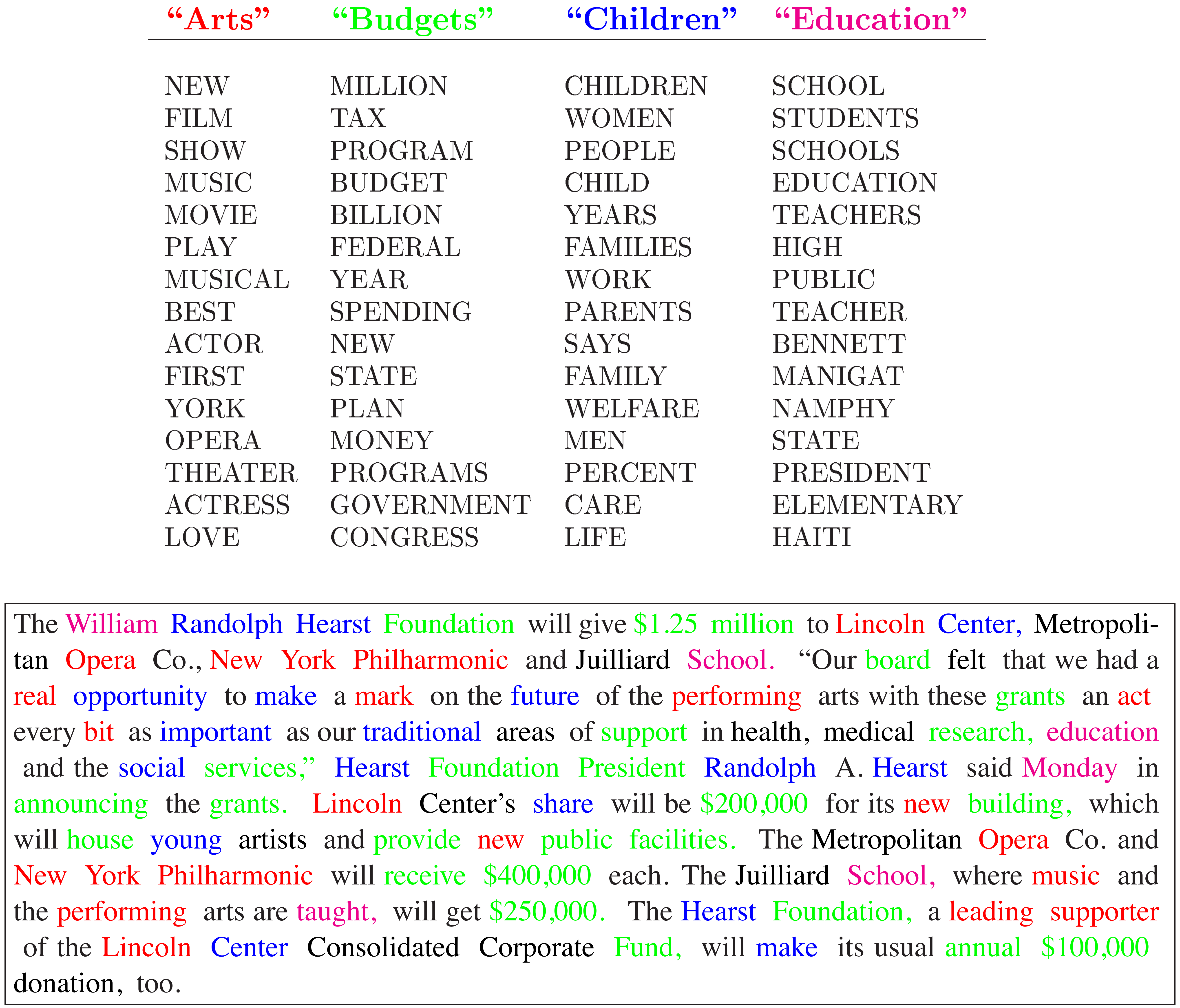
Why do need topics?

- Too **many documents** and we can't read them all!
- Topic models can automatically **categorise** large document collections, so that we can browse through them much more efficiently
- Topic models can be applied on **various corpus collections**, attracting inter-disciplinary interest
  e.g. newspapers, books, social media, health reports
- They can **improve natural language processing tasks**
  e.g. machine translation, word sense disambiguation
- Topics can **improve downstream tasks** in text mining
- Let's see a few **examples**

Topics in news articles

- Latent Dirichlet Allocation (**LDA**) paper (> 50,000 citations)
- Top words from 4 LDA topics
- How different words from these topics (*apologies for the colour coding*) are identified in the text
- Dominant colours → Budgets & Arts seem to be the dominant topics of the paragraph

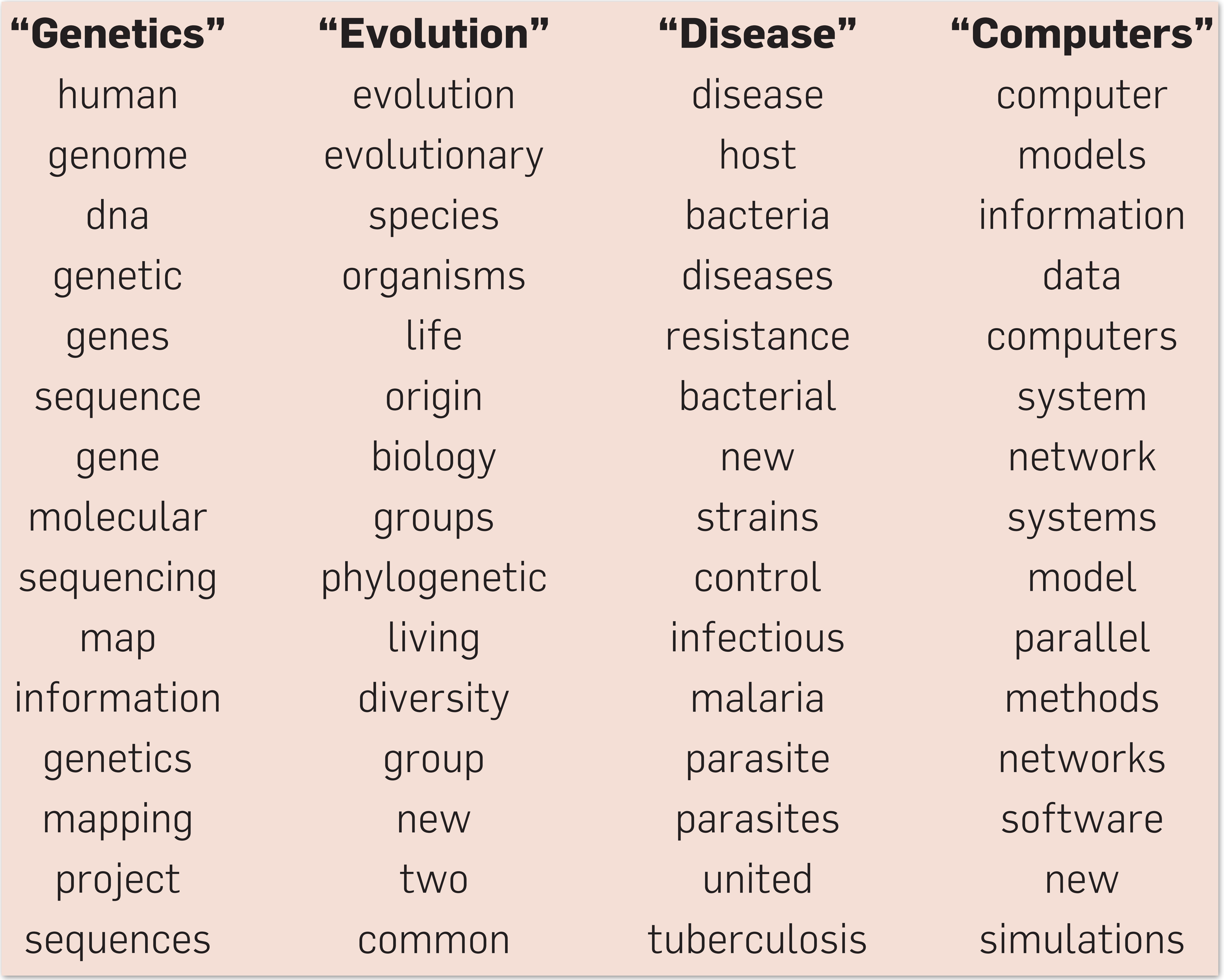
Blei, Ng & Jordan. JMLR, 2003
jmlr.org/papers/volume3/blei03a/blei03a.pdf



"Arts"	"Budgets"	"Children"	"Education"
NEW	MILLION	CHILDREN	SCHOOL
FILM	TAX	WOMEN	STUDENTS
SHOW	PROGRAM	PEOPLE	SCHOOLS
MUSIC	BUDGET	CHILD	EDUCATION
MOVIE	BILLION	YEARS	TEACHERS
PLAY	FEDERAL	FAMILIES	HIGH
MUSICAL	YEAR	WORK	PUBLIC
BEST	SPENDING	PARENTS	TEACHER
ACTOR	NEW	SAYS	BENNETT
FIRST	STATE	FAMILY	MANIGAT
YORK	PLAN	WELFARE	NAMPHY
OPERA	MONEY	MEN	STATE
THEATER	PROGRAMS	PERCENT	PRESIDENT
ACTRESS	GOVERNMENT	CARE	ELEMENTARY
LOVE	CONGRESS	LIFE	HAITI

The William Randolph Hearst Foundation will give $1.25 million to Lincoln Center, Metropolitan Opera Co., New York Philharmonic and Juilliard School. "Our board felt that we had a real opportunity to make a mark on the future of the performing arts with these grants an act every bit as important as our traditional areas of support in health, medical research, education and the social services," Hearst Foundation President Randolph A. Hearst said Monday in announcing the grants. Lincoln Center's share will be $200,000 for its new building, which will house young artists and provide new public facilities. The Metropolitan Opera Co. and New York Philharmonic will receive $400,000 each. The Juilliard School, where music and the performing arts are taught, will get $250,000. The Hearst Foundation, a leading supporter of the Lincoln Center Consolidated Corporate Fund, will make its usual annual $100,000 donation, too.

17,000 articles from the journal "Science"



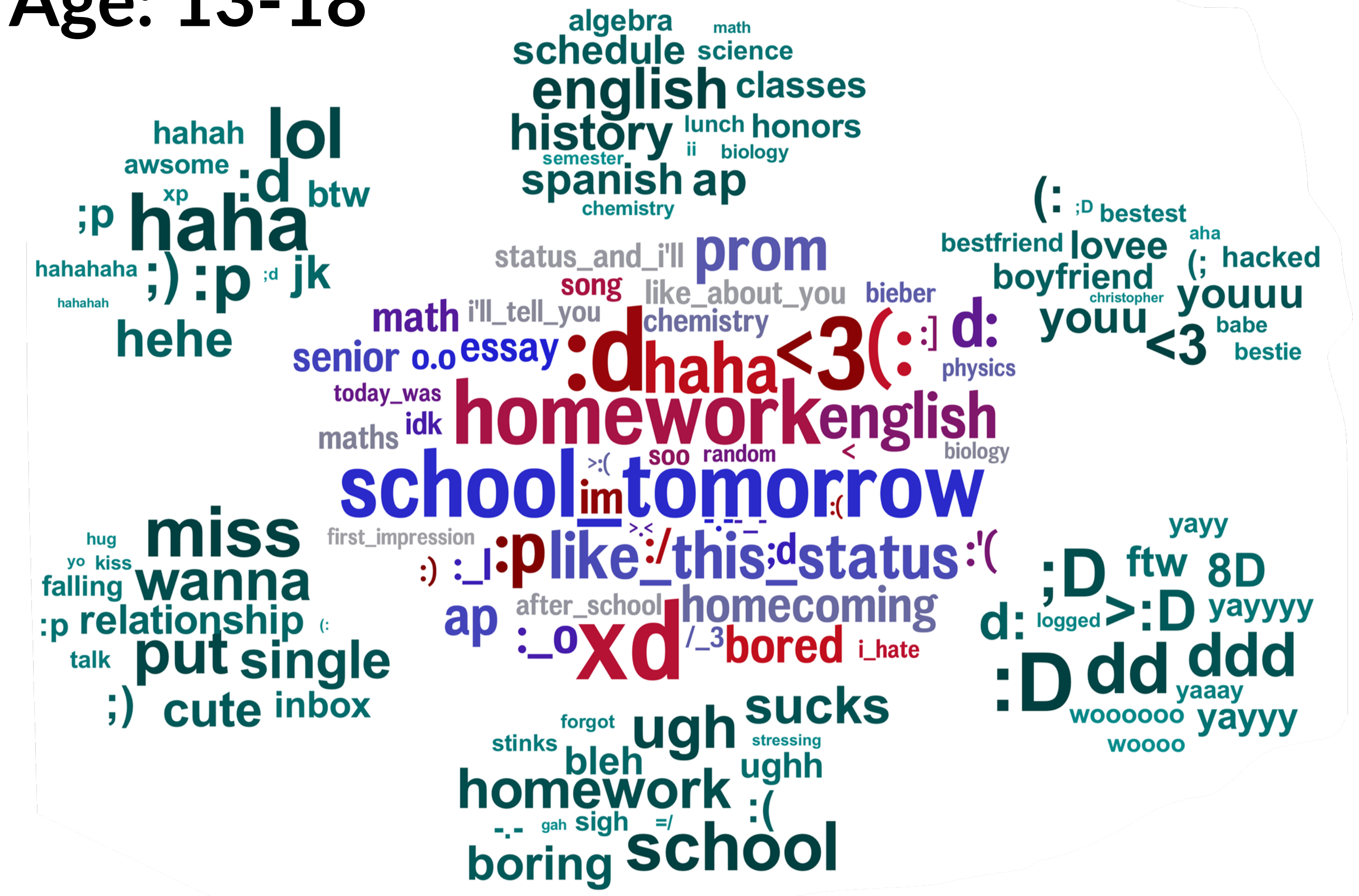
"Genetics"	"Evolution"	"Disease"	"Computers"
human	evolution	disease	computer
genome	evolutionary	host	models
dna	species	bacteria	information
genetic	organisms	diseases	data
genes	life	resistance	computers
sequence	origin	bacterial	system
gene	biology	new	network
molecular	groups	strains	systems
sequencing	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

- Different source data, different topics and language (words)
- more scientific / technical language

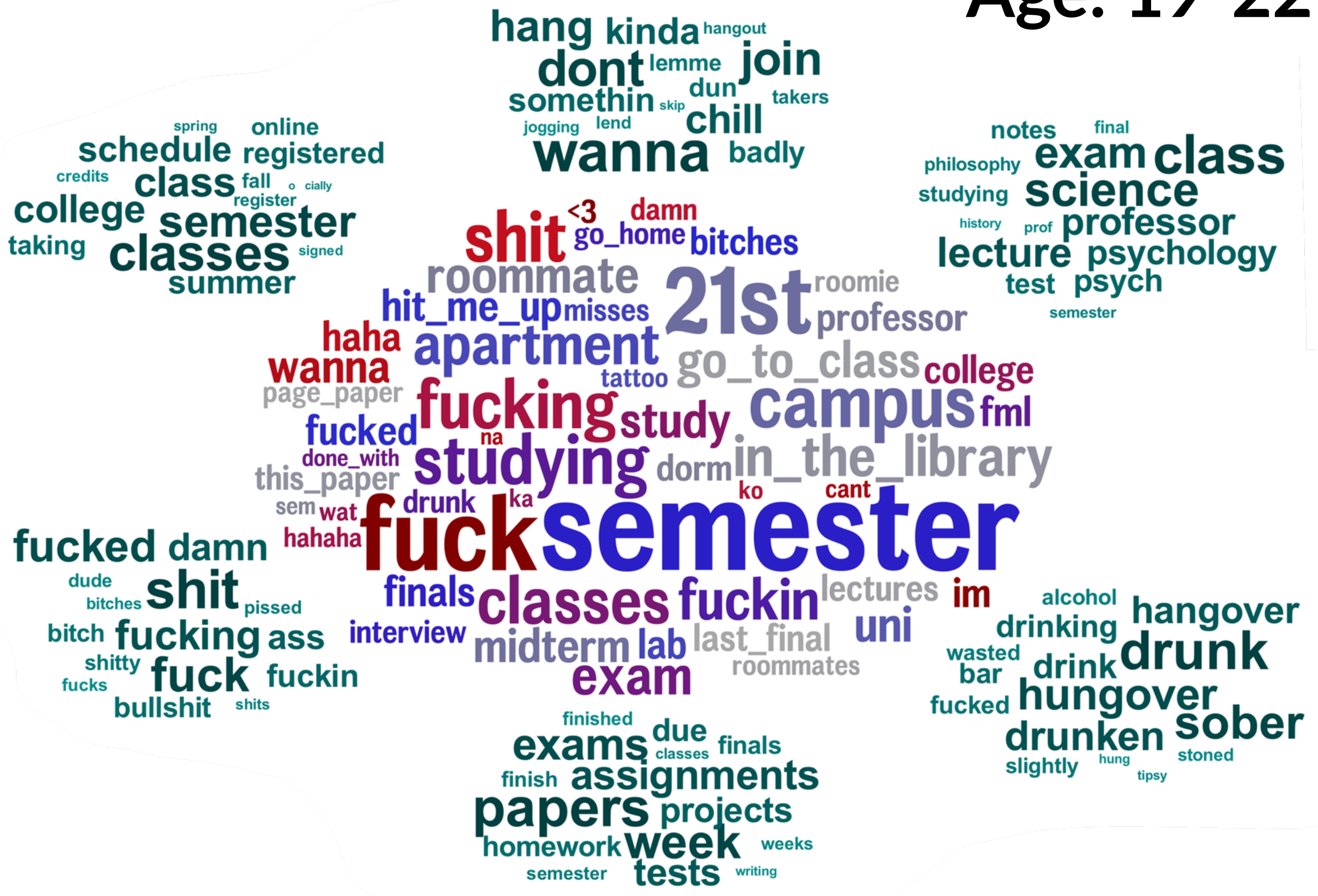
Blei. CACM, 2012
doi.org/10.1145/2133806.2133826

Age-group topics on Facebook

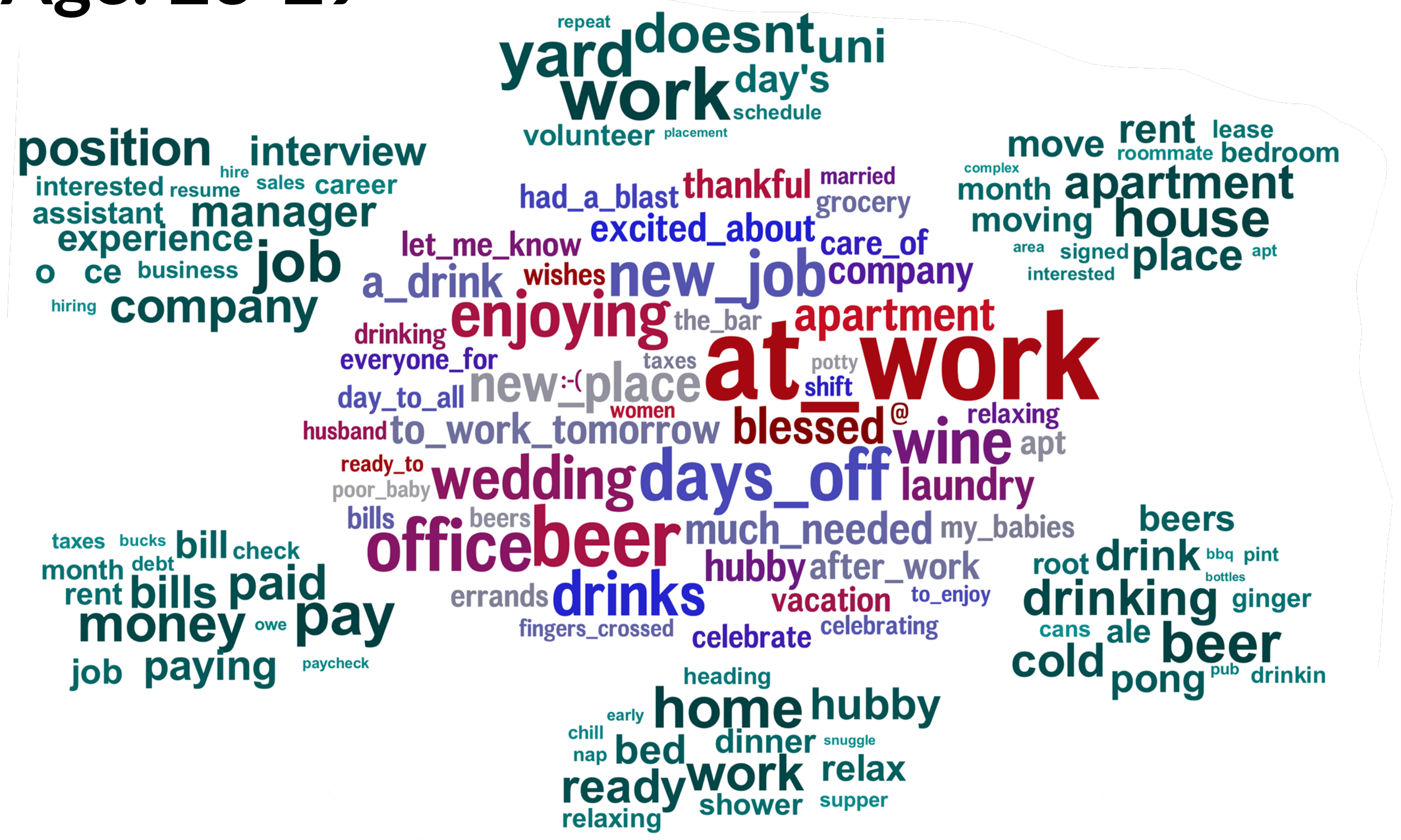
Age: 13-18



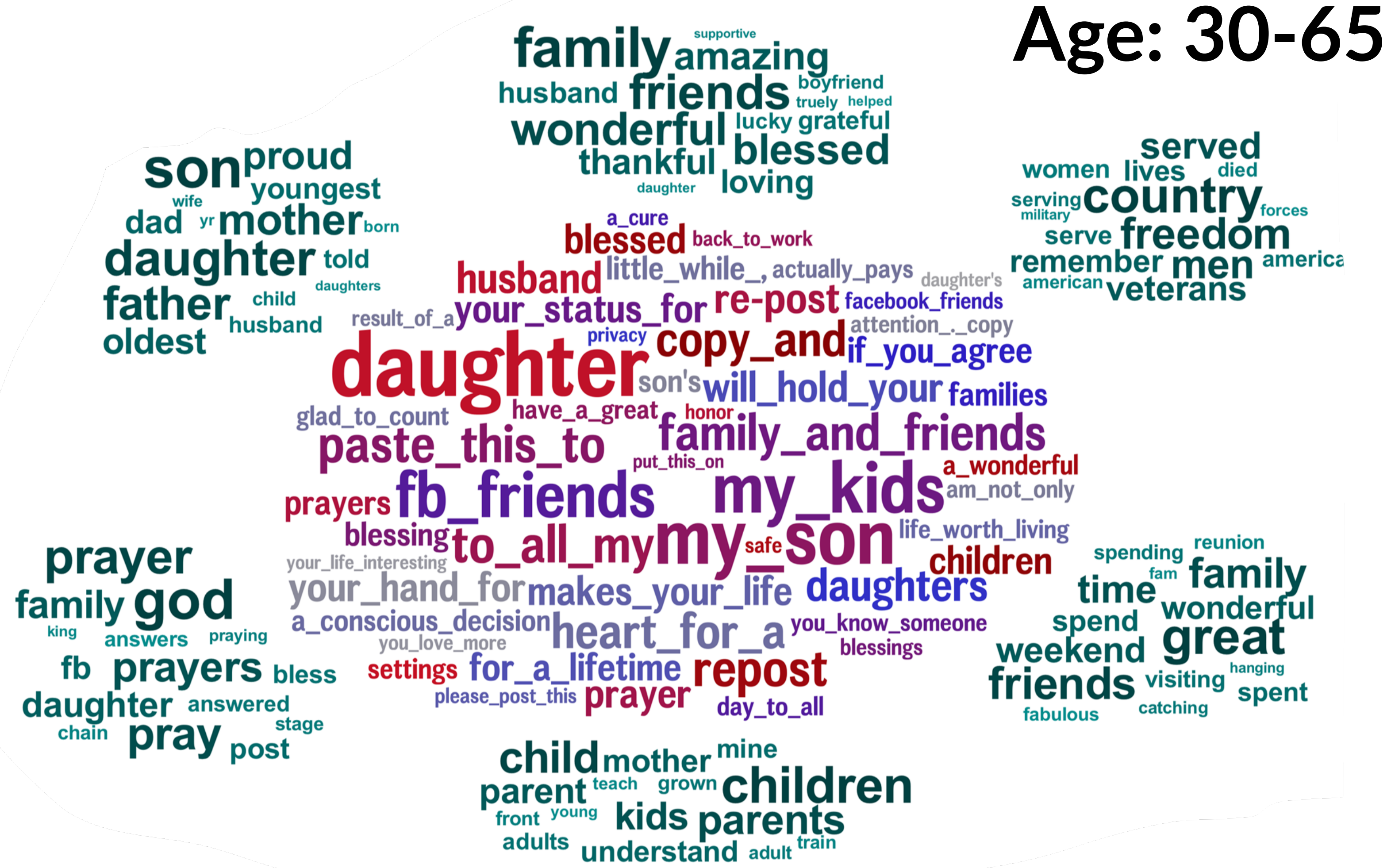
Age: 19-22



Age: 23-29

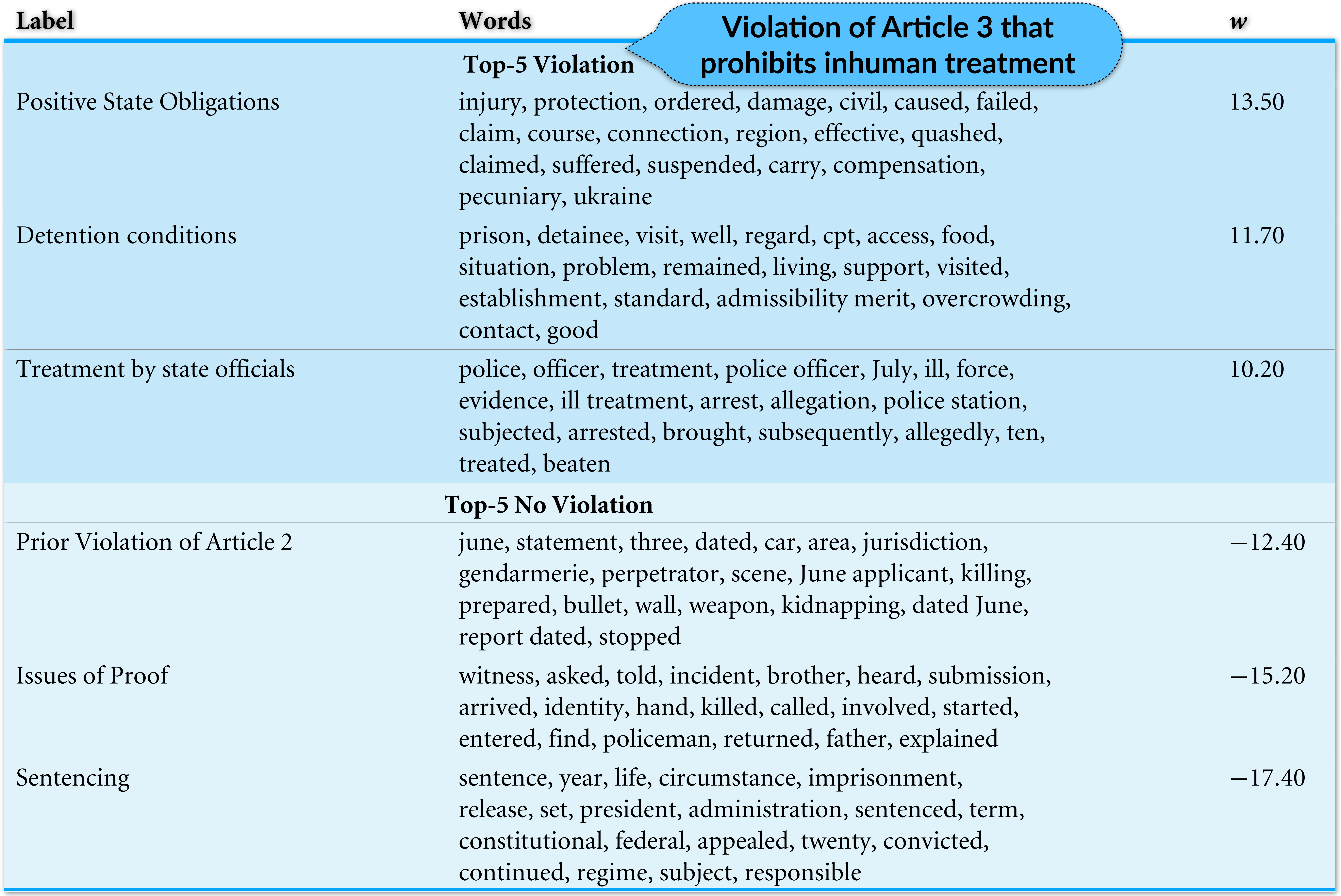


Age: 30-65



Schwartz et al. PLOS ONE, 2013
doi.org/10.1371/journal.pone.0073791

Predicting judicial decisions

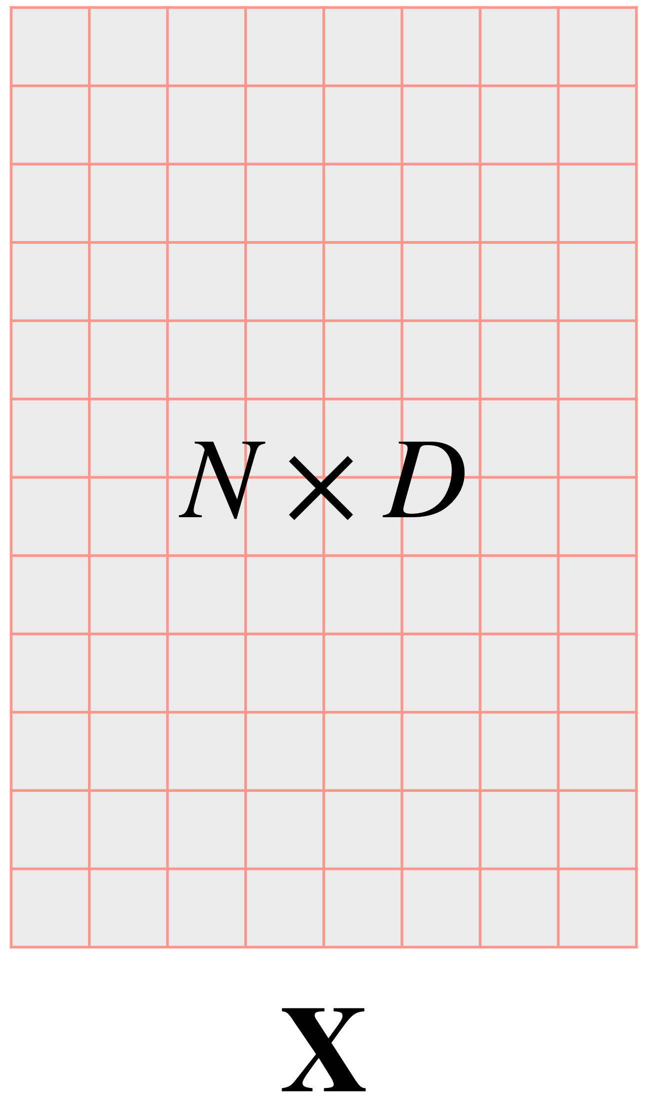


Label	Words	*w*
	Top-5 Violation	
Positive State Obligations	injury, protection, ordered, damage, civil, caused, failed, claim, course, connection, region, effective, quashed, claimed, suffered, suspended, carry, compensation, pecuniary, ukraine	13.50
Detention conditions	prison, detainee, visit, well, regard, cpt, access, food, situation, problem, remained, living, support, visited, establishment, standard, admissibility merit, overcrowding, contact, good	11.70
Treatment by state officials	police, officer, treatment, police officer, July, ill, force, evidence, ill treatment, arrest, allegation, police station, subjected, arrested, brought, subsequently, allegedly, ten, treated, beaten	10.20
	Top-5 No Violation	
Prior Violation of Article 2	june, statement, three, dated, car, area, jurisdiction, gendarmerie, perpetrator, scene, June applicant, killing, prepared, bullet, wall, weapon, kidnapping, dated June, report dated, stopped	−12.40
Issues of Proof	witness, asked, told, incident, brother, heard, submission, arrived, identity, hand, killed, called, involved, started, entered, find, policeman, returned, father, explained	−15.20
Sentencing	sentence, year, life, circumstance, imprisonment, release, set, president, administration, sentenced, term, constitutional, federal, appealed, twenty, convicted, continued, regime, subject, responsible	−17.40

Violation of Article 3 that prohibits inhuman treatment

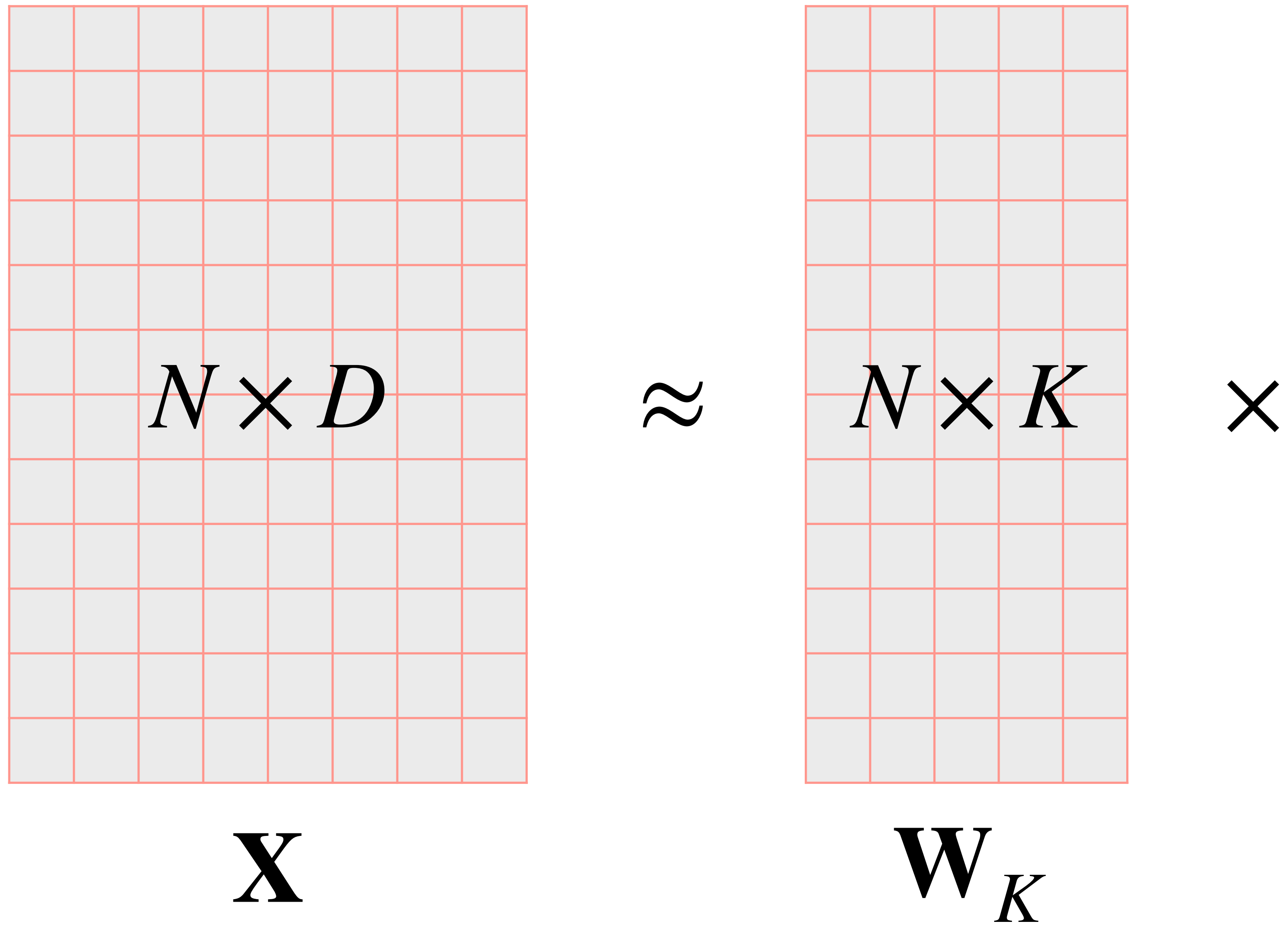
Aletras, Tsarapatsanis, Preotiuc, Lampos. PeerJ Computer Science, 2016
doi.org/10.7717/peerj-cs.93

Latent Semantic Analysis (or Indexing) — LSA



Singular Value Decomposition (SVD; truncated) on the term-document matrix $\mathbf{X}$ representing the frequency of N terms (or n-grams) in D documents

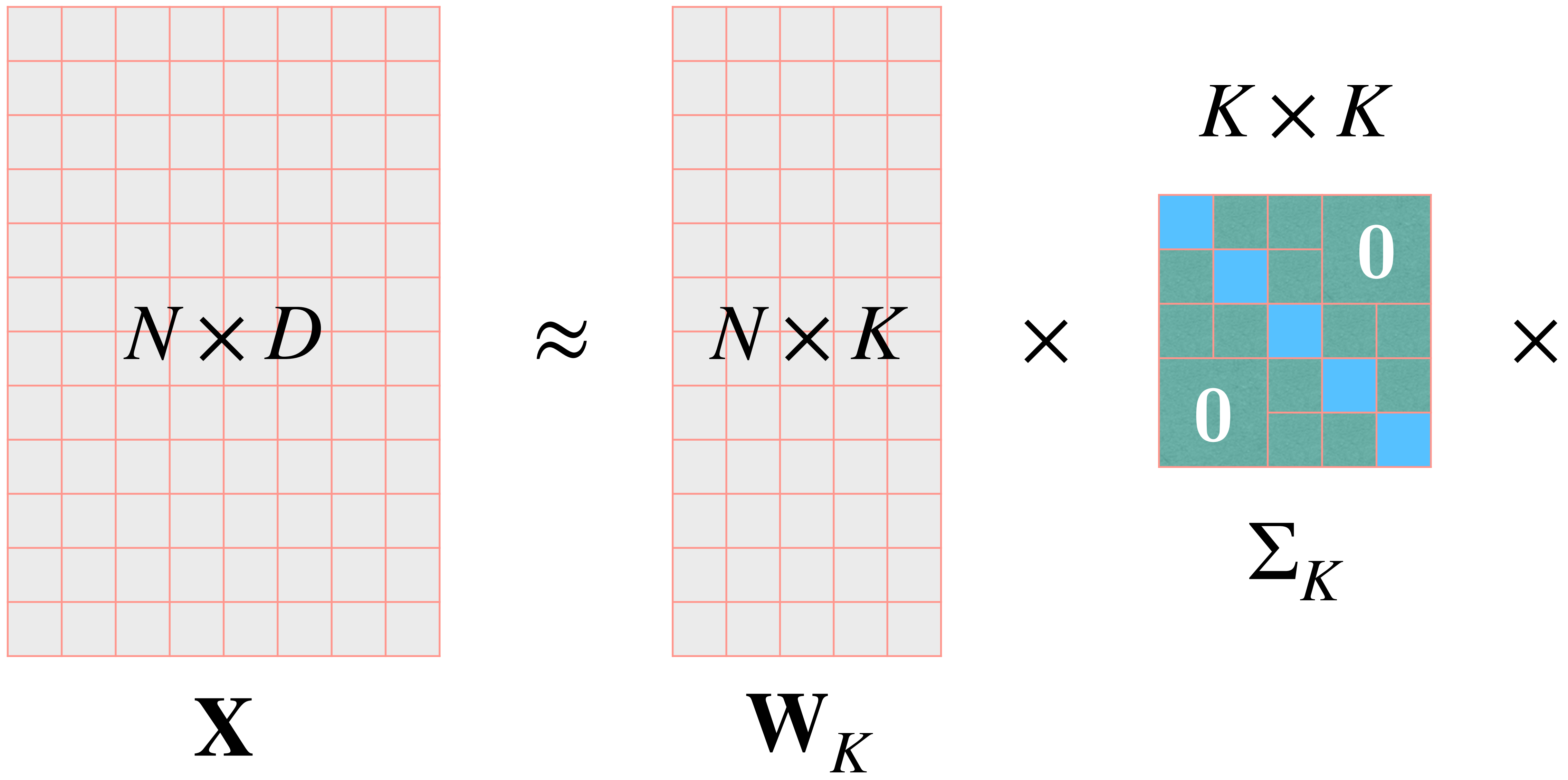
Latent Semantic Analysis (or Indexing) — LSA



Singular Value Decomposition (SVD; truncated) on the term-document matrix $\mathbf{X}$ representing the frequency of N terms (or n-grams) in D documents

$\mathbf{W}_K$: each topic’s (K topics) distribution over N terms

Latent Semantic Analysis (or Indexing) — LSA

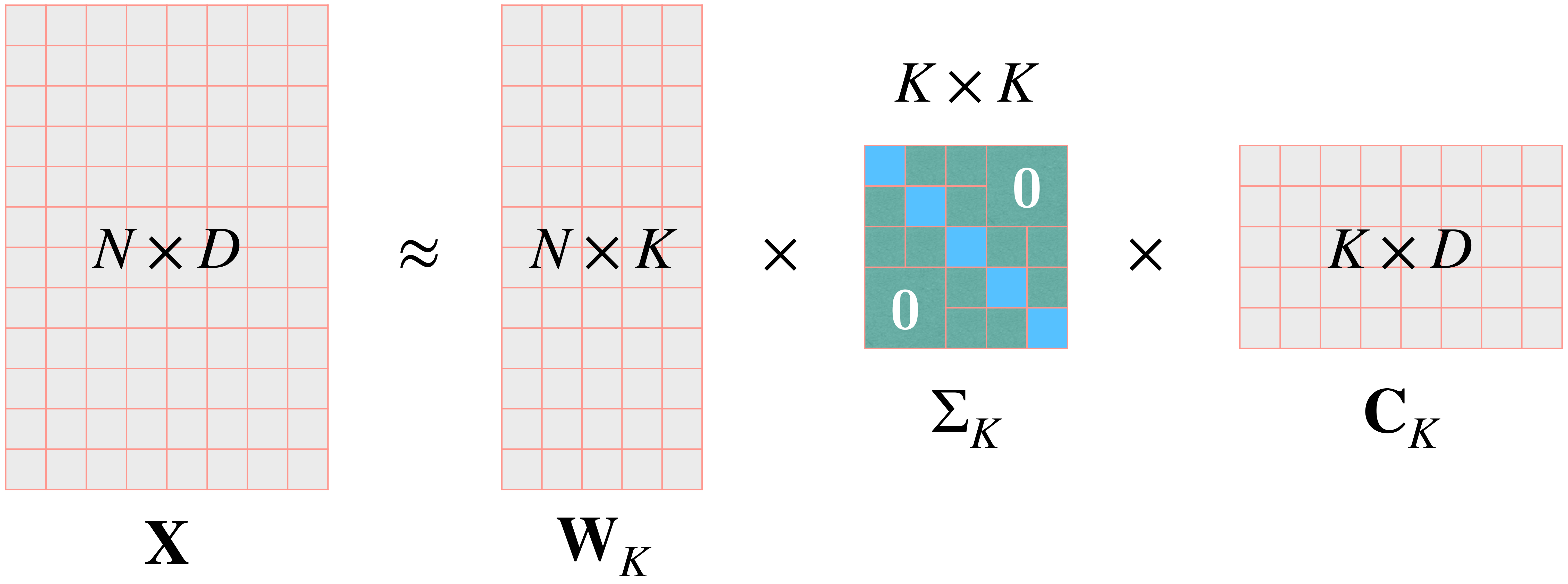


Singular Value Decomposition (SVD; truncated) on the term-document matrix $\mathbf{X}$ representing the frequency of N terms (or n-grams) in D documents

$\mathbf{W}_K$: each topic's (K topics) distribution over N terms

Σ_K : diagonal matrix, can be seen as a topic importance / weight

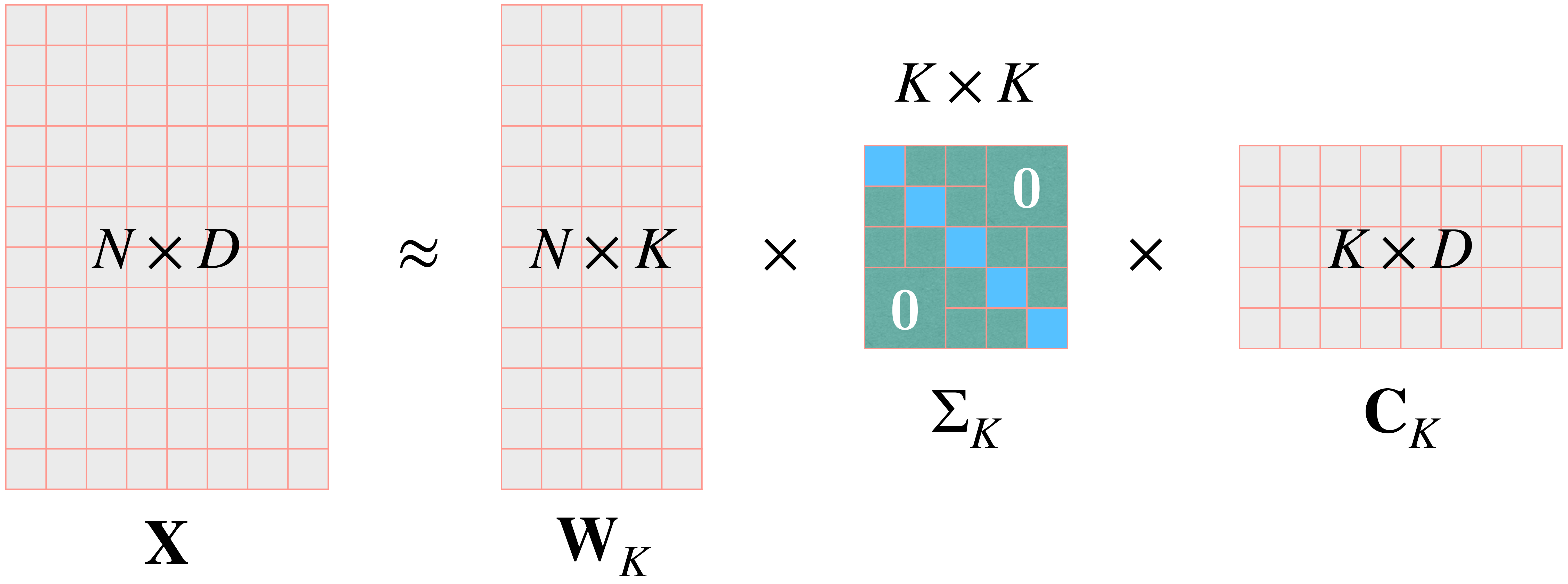
Latent Semantic Analysis (or Indexing) — LSA



Singular Value Decomposition (SVD; truncated) on the term-document matrix $\mathbf{X}$ representing the frequency of N terms (or n-grams) in D documents

$\mathbf{W}_K$: each topic's (K topics) distribution over N terms
Σ_K : diagonal matrix, can be seen as a topic importance / weight
$\mathbf{C}_K$: each document's (D documents) distribution over K topics

Latent Semantic Analysis (or Indexing) — LSA



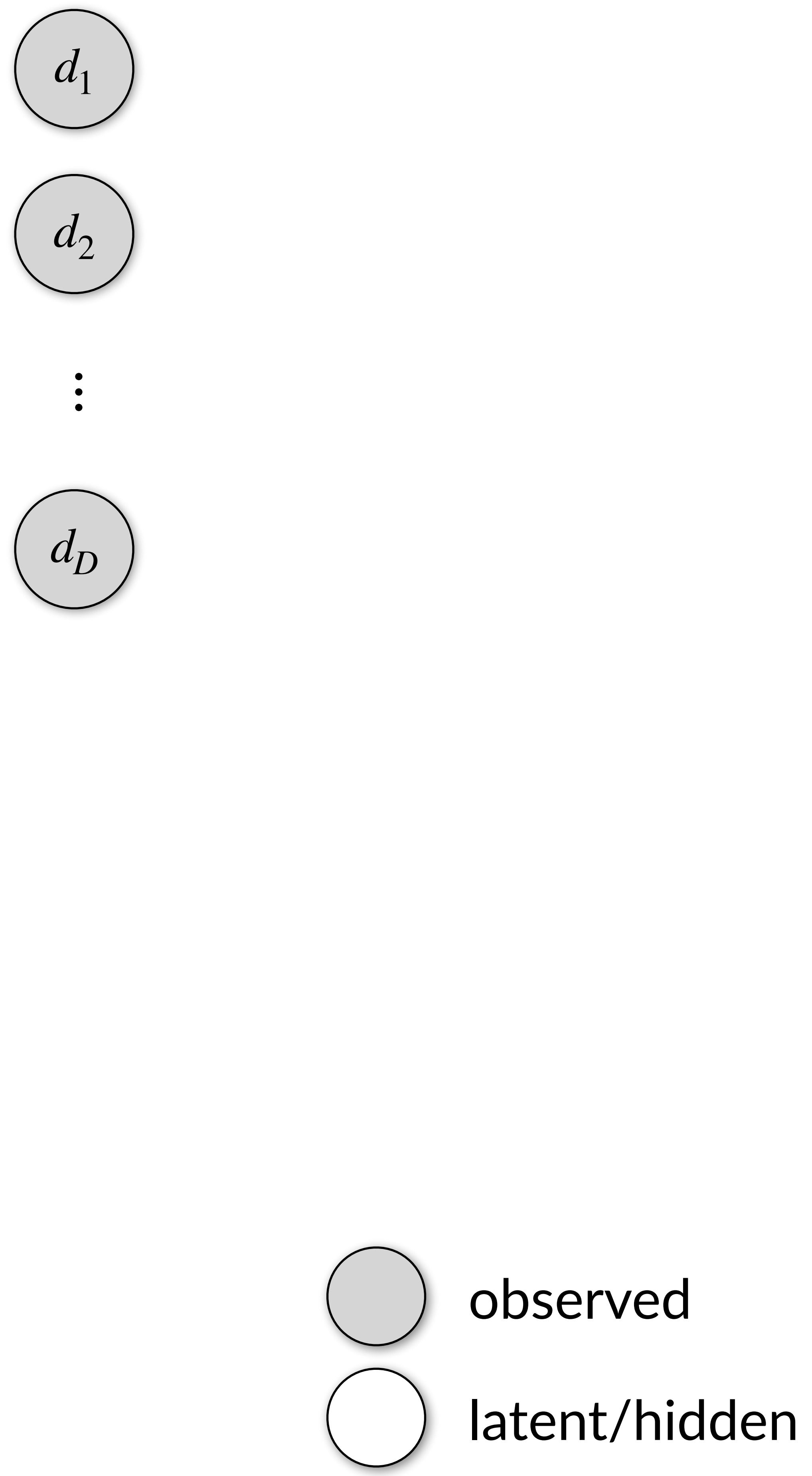
Disadvantages

— SVD has a significant computational cost $\approx \mathcal{O}\left(NDK^2\right)$

— No intuition about the origin of the topics (*brute force method*)

$\longrightarrow$ ***probabilistic topic models!***

Probabilistic LSA — pLSA

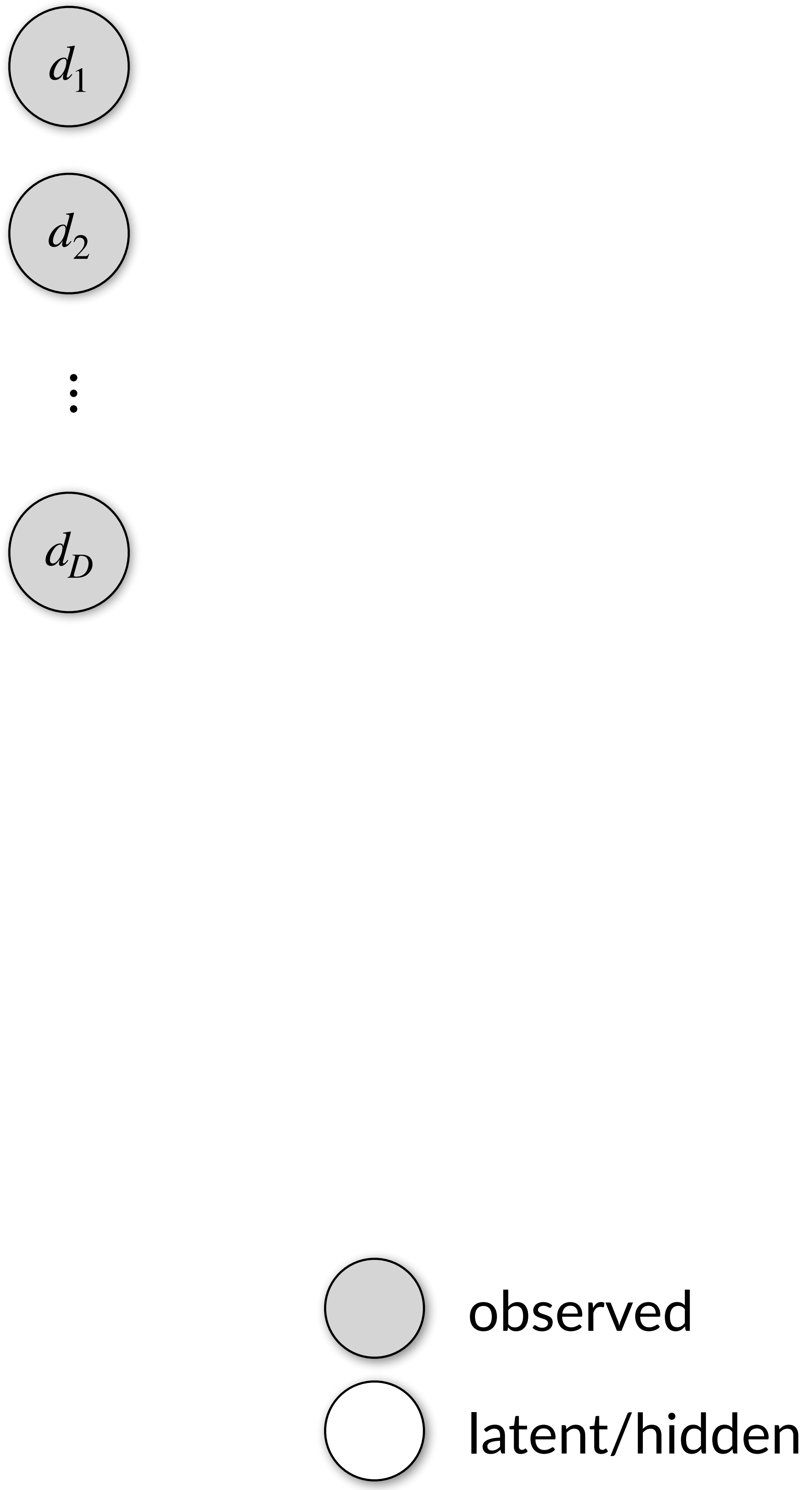


Probabilistic topic models try to explain how the documents in our collection were generated

$\longrightarrow$ ***generative story*** behind the derived topic models

For all j documents (1 to D):
- Select a document d_j with probability $p(d_j)$
- Choose a mixture of K topics $\boldsymbol{\theta}_j$ for document d_j
- For each word position i (1 to N) in the document d_j:
  - Choose a topic z_k with probability $p(z_k \mid d_j)$
  - Choose a term/word w_i with probability $p(w_i \mid z_k)$

Probabilistic LSA — pLSA

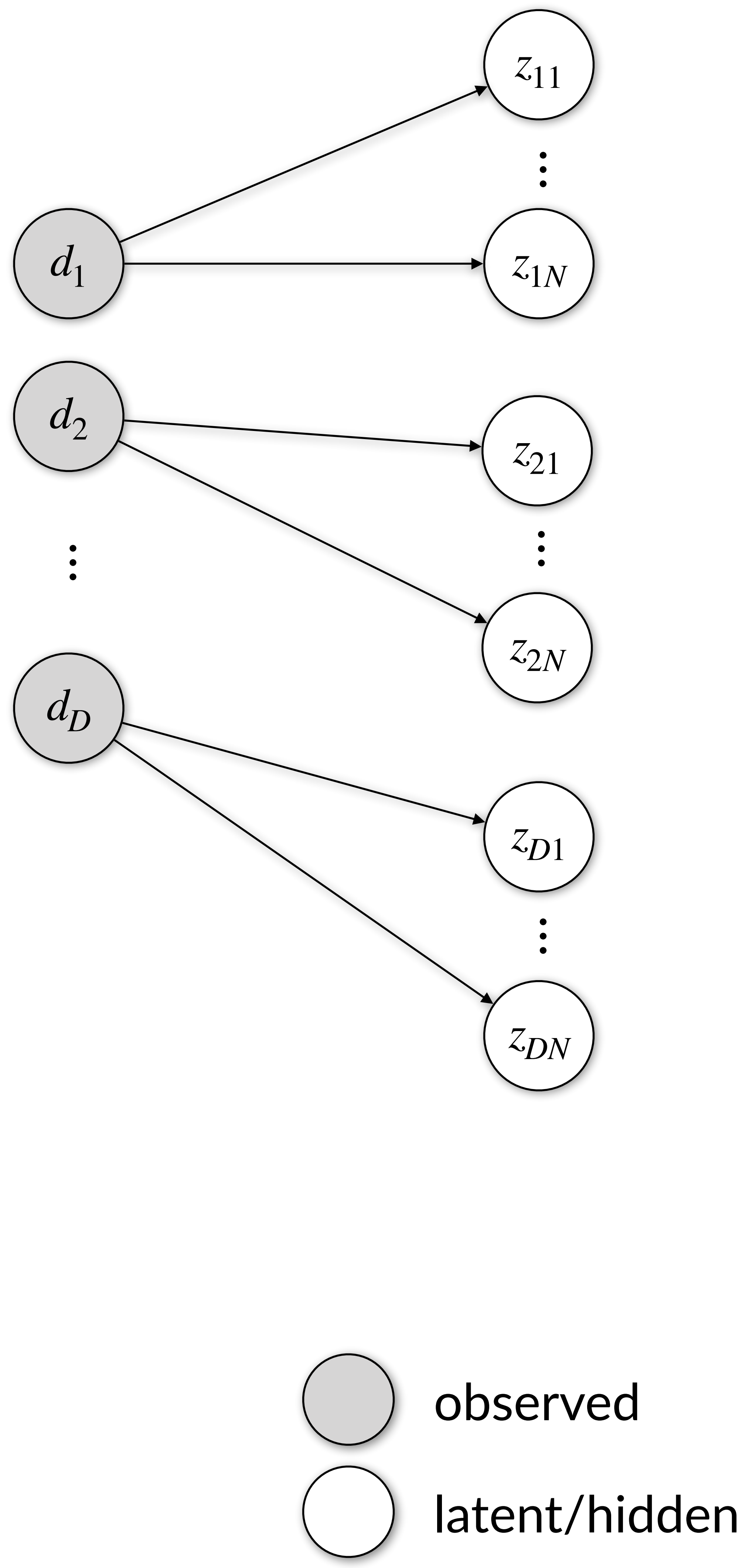


Probabilistic topic models try to explain how the documents in our collection were generated

⟶ ***generative story*** behind the derived topic models

For all j documents (1 to D):
— Select a document d_j with probability $p(d_j)$
— Choose a mixture of K topics $\boldsymbol{\theta}_j$ for document d_j
— For each word position i (1 to N) in the document d_j:
  — Choose a topic z_k with probability $p(z_k \mid d_j)$
  — Choose a term/word w_i with probability $p(w_i \mid z_k)$

Probabilistic LSA — pLSA

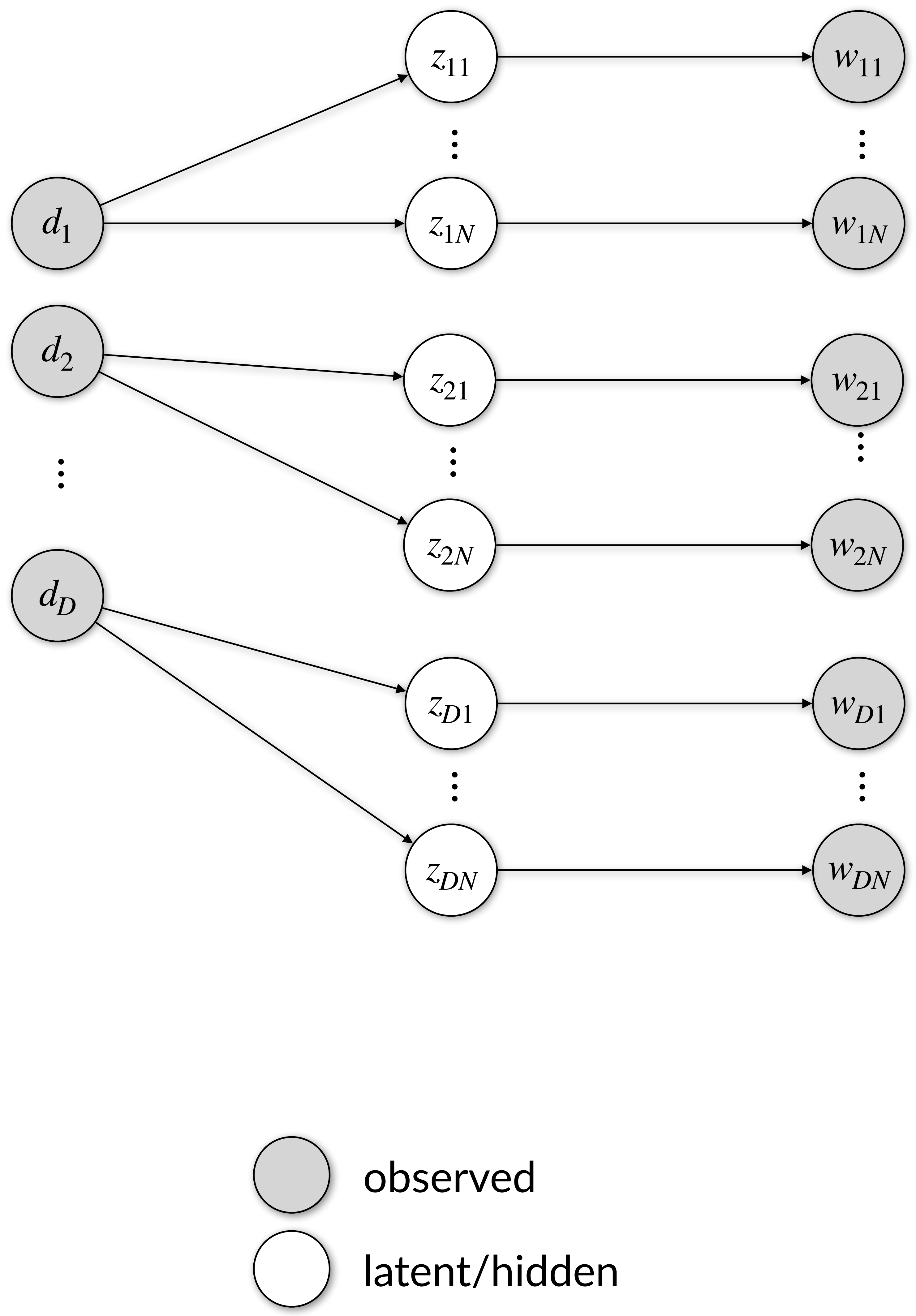


Probabilistic topic models try to explain how the documents in our collection were generated

⟶ ***generative story*** behind the derived topic models

For all j documents (1 to D):
- Select a document d_j with probability $p(d_j)$
- Choose a mixture of K topics $\boldsymbol{\theta}_j$ for document d_j
- For each word position i (1 to N) in the document d_j:
  - Choose a topic z_k with probability $p(z_k \mid d_j)$
  - Choose a term/word w_i with probability $p(w_i \mid z_k)$

Probabilistic LSA — pLSA



Probabilistic topic models try to explain how the documents in our collection were generated

⟶ ***generative story*** behind the derived topic models

For all j documents (1 to D):
— Select a document d_j with probability $p(d_j)$
— Choose a mixture of K topics $\boldsymbol{\theta}_j$ for document d_j
— For each word position i (1 to N) in the document d_j:
 — Choose a topic z_k with probability $p(z_k \,|\, d_j)$
 — Choose a term/word w_i with probability $p(w_i \,|\, z_k)$

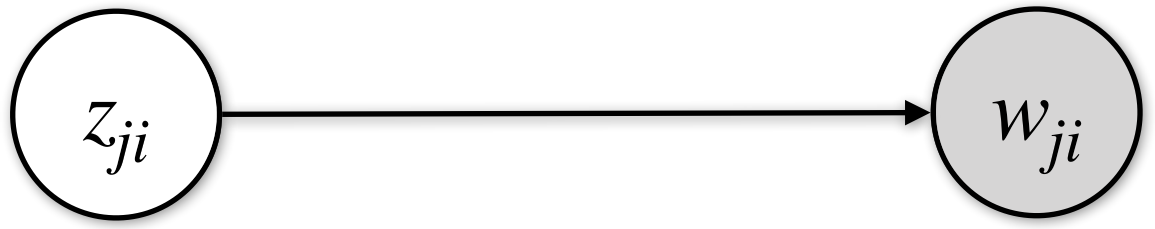
Generative story: *the topic distribution that characterises a document in our collection determines which words should exist in it*

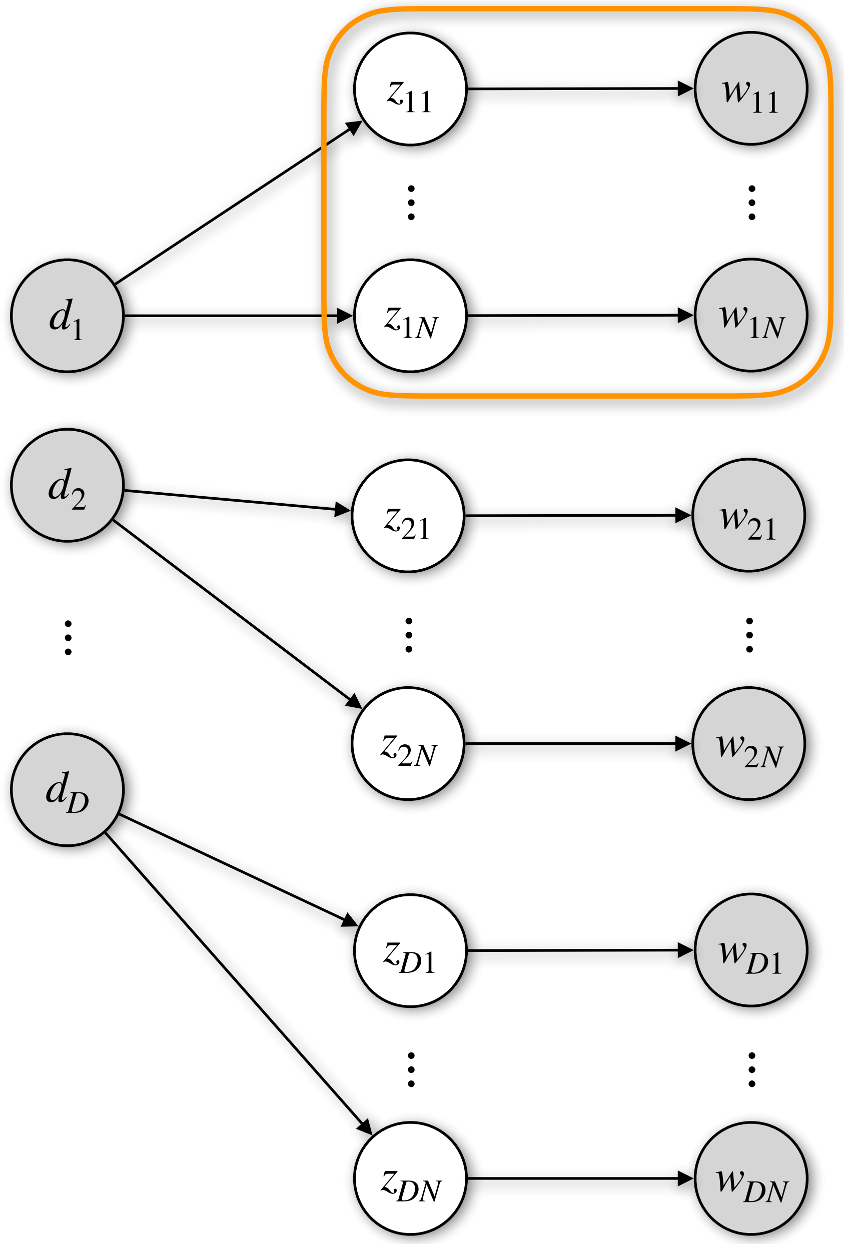
Probabilistic LSA — pLSA

Plate notation

● observed

○ latent/hidden





For all j documents (1 to D):

— Select a document d_j with probability $p(d_j)$

— Choose a mixture of K topics $\boldsymbol{\theta}_j$ for document d_j

— For each word position i (1 to N) in the document d_j:

  — Choose a topic z_k with probability $p(z_k \mid d_j)$

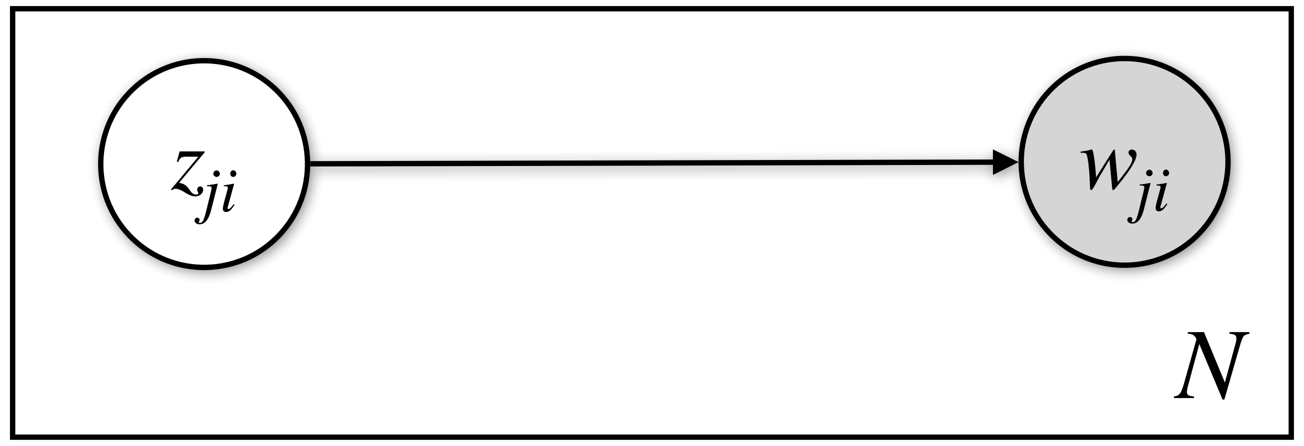
  — Choose a term/word w_i with probability $p(w_i \mid z_k)$

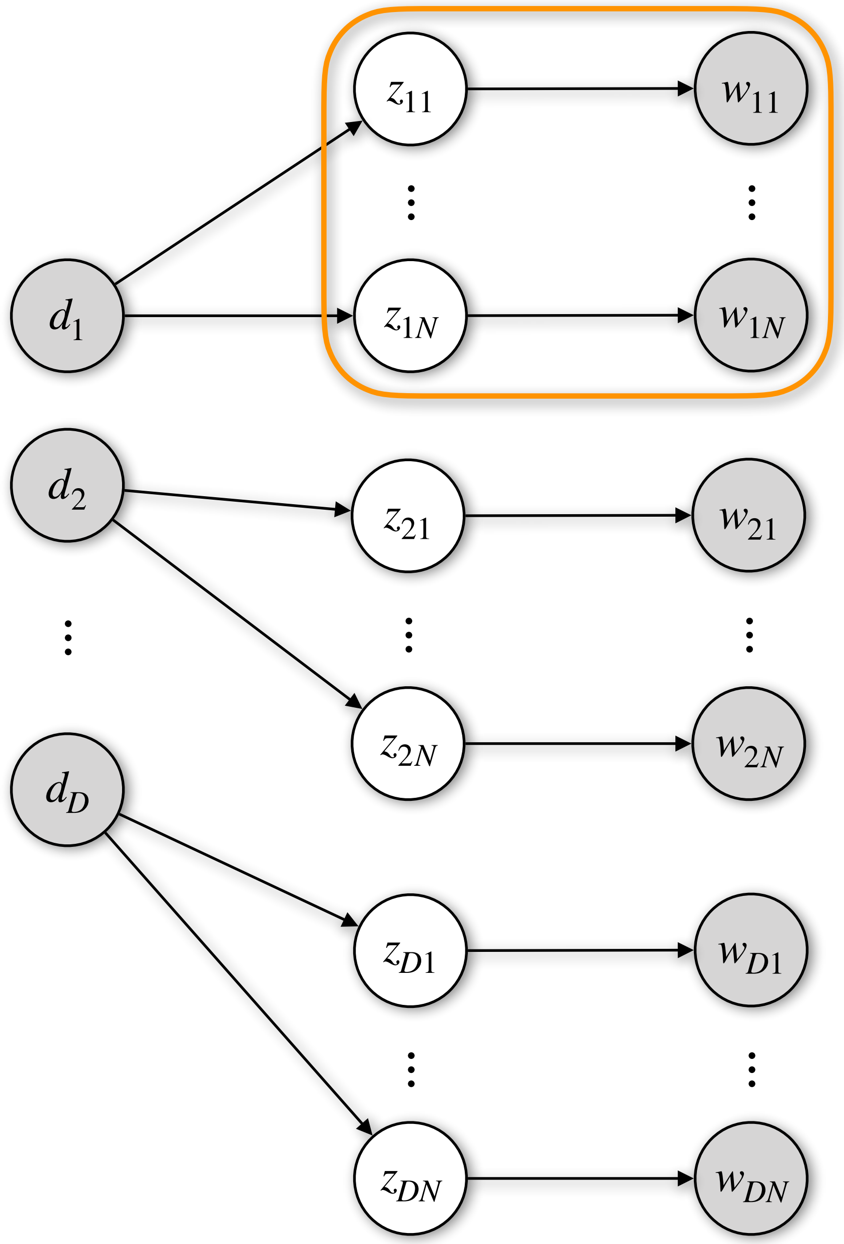
Probabilistic LSA — pLSA

● observed

○ latent/hidden

Plate notation





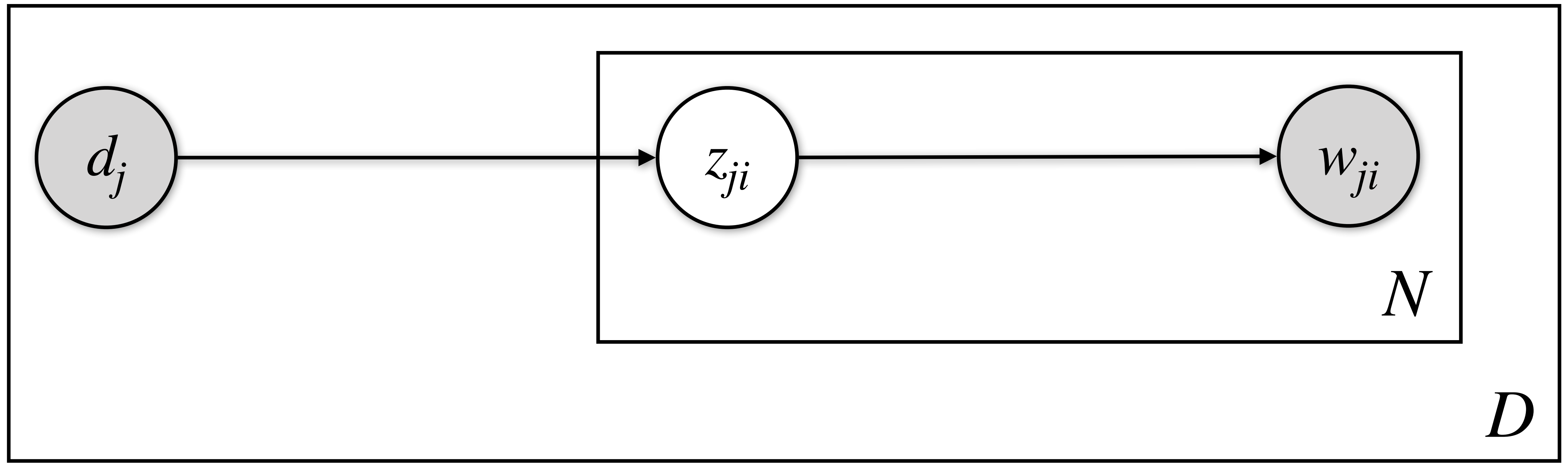
For all j documents (1 to D):
- Select a document d_j with probability $p(d_j)$
- Choose a mixture of K topics $\boldsymbol{\theta}_j$ for document d_j
- For each word position i (1 to N) in the document d_j:
  - Choose a topic z_k with probability $p(z_k \mid d_j)$
  - Choose a term/word w_i with probability $p(w_i \mid z_k)$

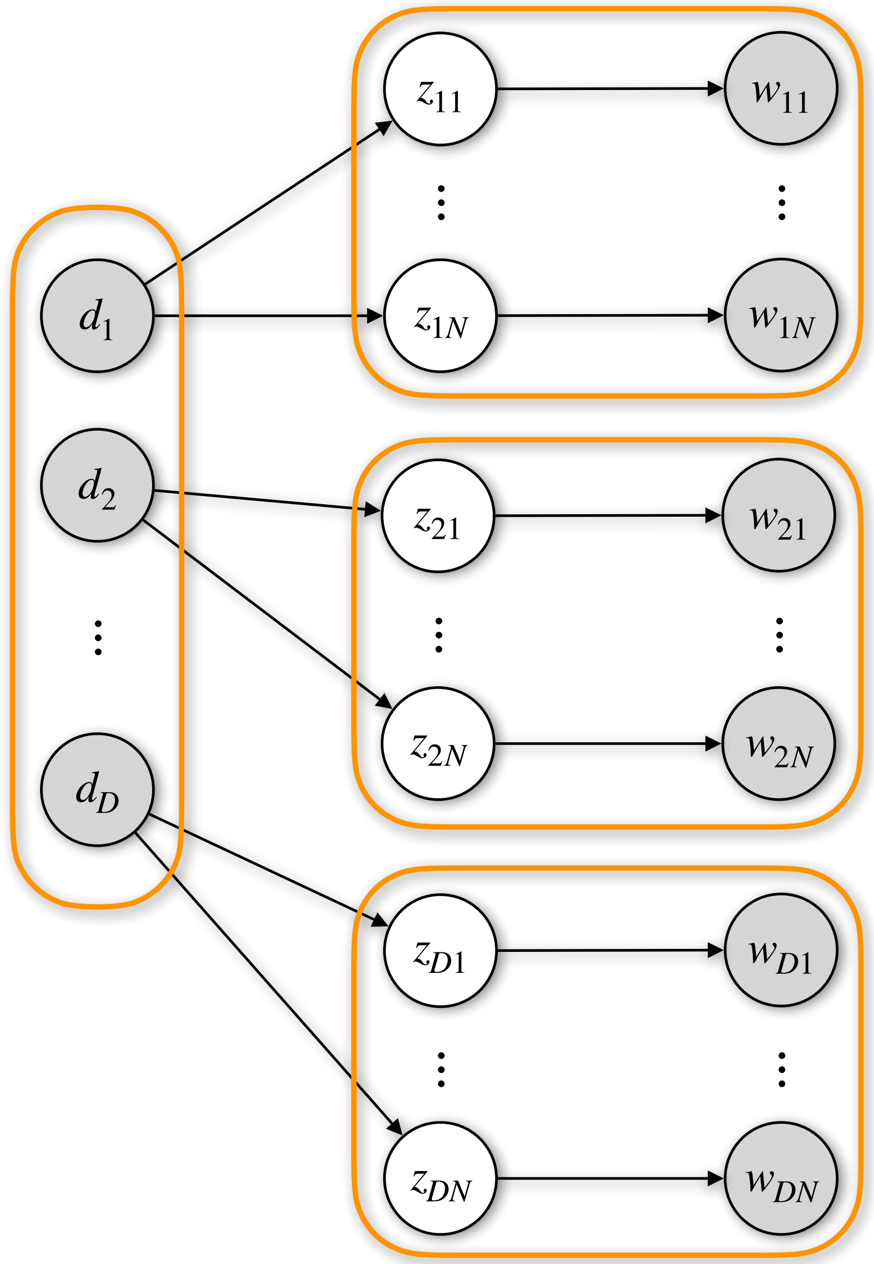
Probabilistic LSA — pLSA

● observed

○ latent/hidden

Plate notation





For all j documents (1 to D):

– Select a document d_j with probability $p(d_j)$

– Choose a mixture of K topics $\boldsymbol{\theta}_j$ for document d_j

– For each word position i (1 to N) in the document d_j:

  – Choose a topic z_k with probability $p(z_k \mid d_j)$

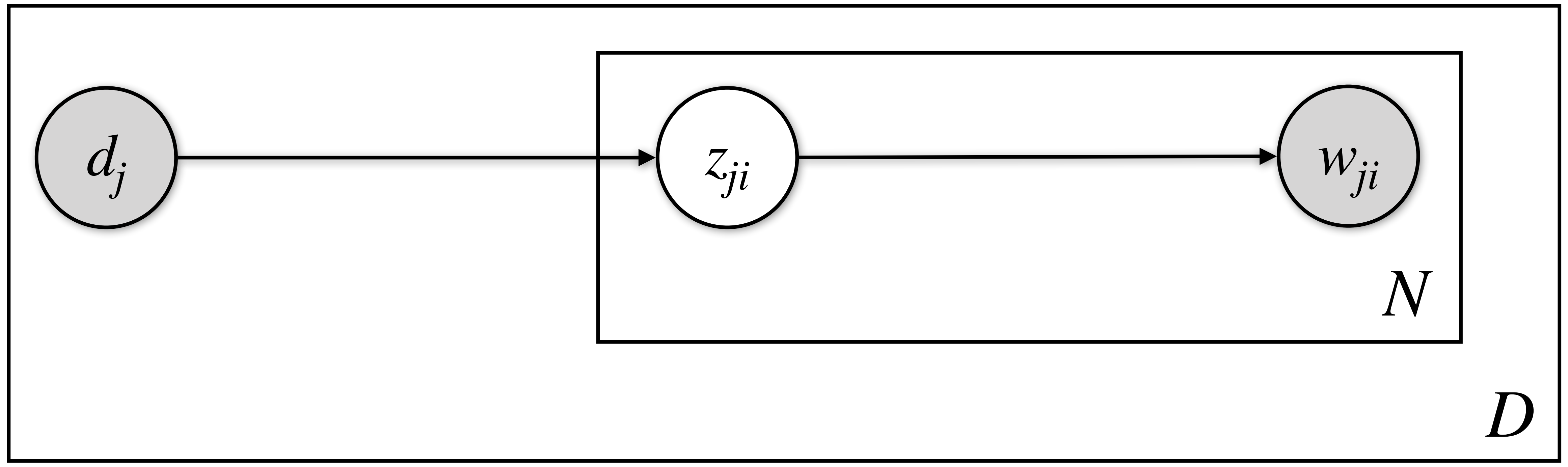
  – Choose a term/word w_i with probability $p(w_i \mid z_k)$

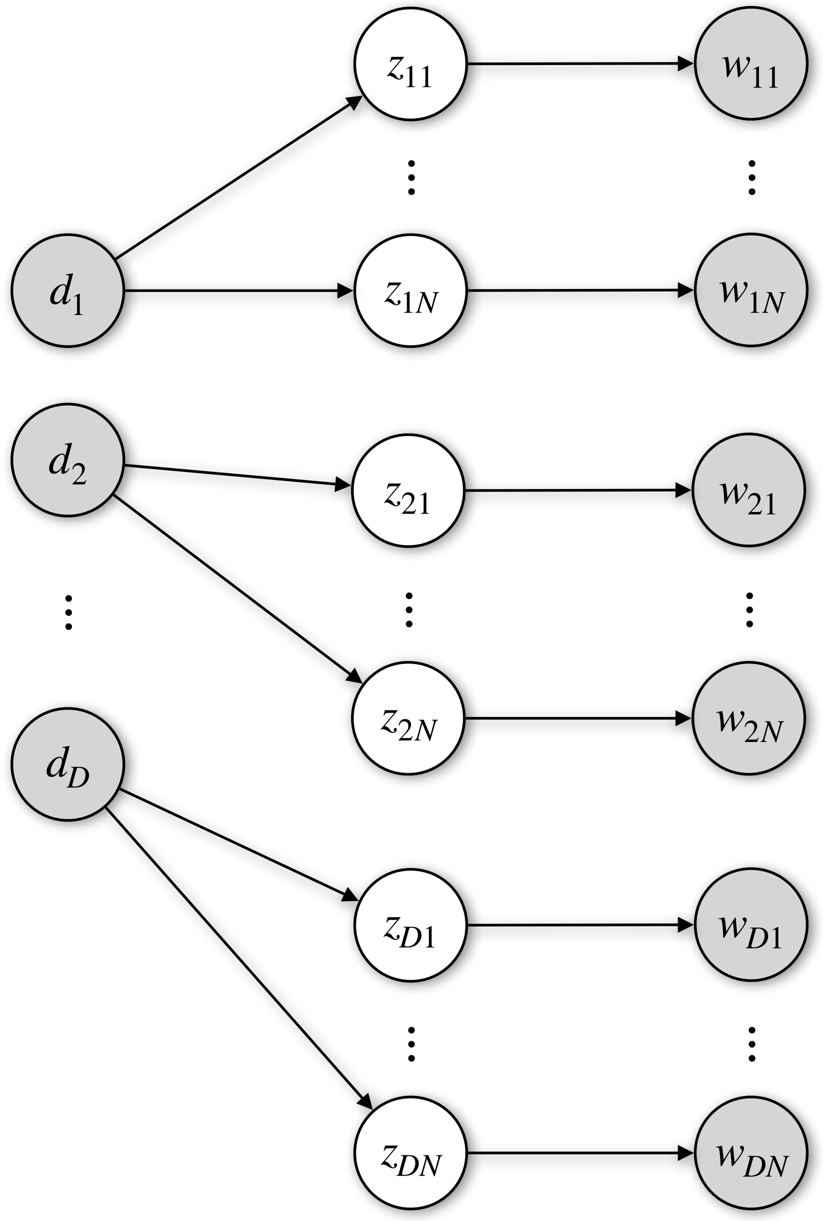
Probabilistic LSA — pLSA

● observed

○ latent/hidden

Plate notation





joint probability distribution

$$p(\mathbf{d}, \mathbf{W}) = \prod_{j=1}^{D} p(d_j) \prod_{i=1}^{N} \sum_{k=1}^{K} p(z_{ji} = k \,|\, d_j)\, p(w_{ji} \,|\, z_{ji} = k)$$

For all j documents (1 to D):

— Select a document d_j with probability $p(d_j)$

— Choose a mixture of K topics $\boldsymbol{\theta}_j$ for document d_j

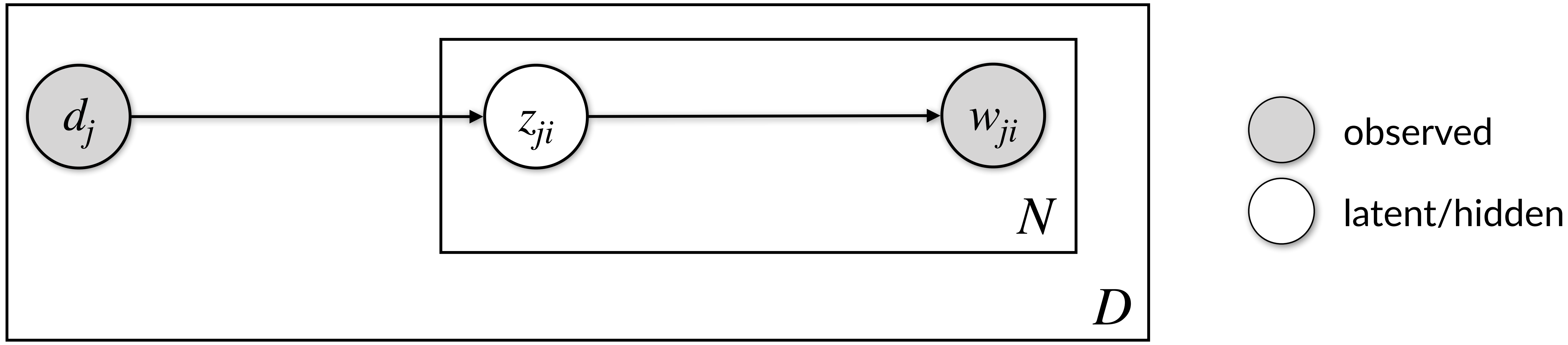
— For each word position i (1 to N) in the document d_j:

  — Choose a topic z_k with probability $p(z_k \,|\, d_j)$

  — Choose a term/word w_i with probability $p(w_i \,|\, z_k)$

Probabilistic LSA — pLSA

Plate notation



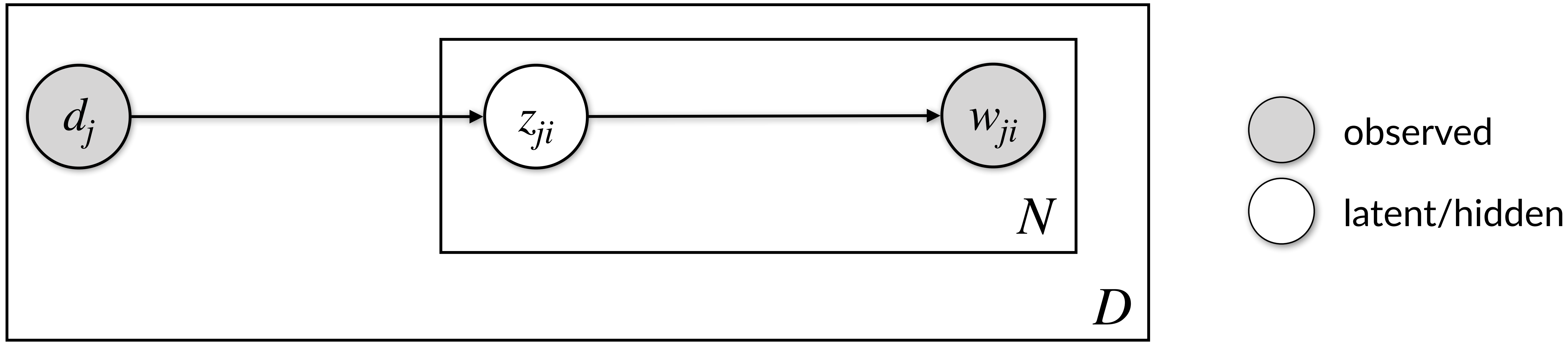
Assumptions: In a document d_j, a word w_{ji} is generated from a single topic z_{ji} from the K assumed ones, and given that topic, the word is independent of all of the other words in that document.

$$p(d_j, w_i) = p(d_j)\, p(w_i \,|\, d_j) = p(d_j) \sum_{k=1}^{K} p(z = k \,|\, d_j)\, p(w_i \,|\, z = k)$$

Joint probability distribution for d_j and w_i
(*single word in the document*)

Probabilistic LSA — pLSA

Plate notation



Assumptions: In a document d_j, a word w_{ji} is generated from a single topic z_{ji} from the K assumed ones, and given that topic, the word is independent of all of the other words in that document.

$$p(d_j, w_i) = p(d_j)\, p(w_i \,|\, d_j) = p(d_j) \sum_{k=1}^{K} p(z = k \,|\, d_j)\, p(w_i \,|\, z = k)$$

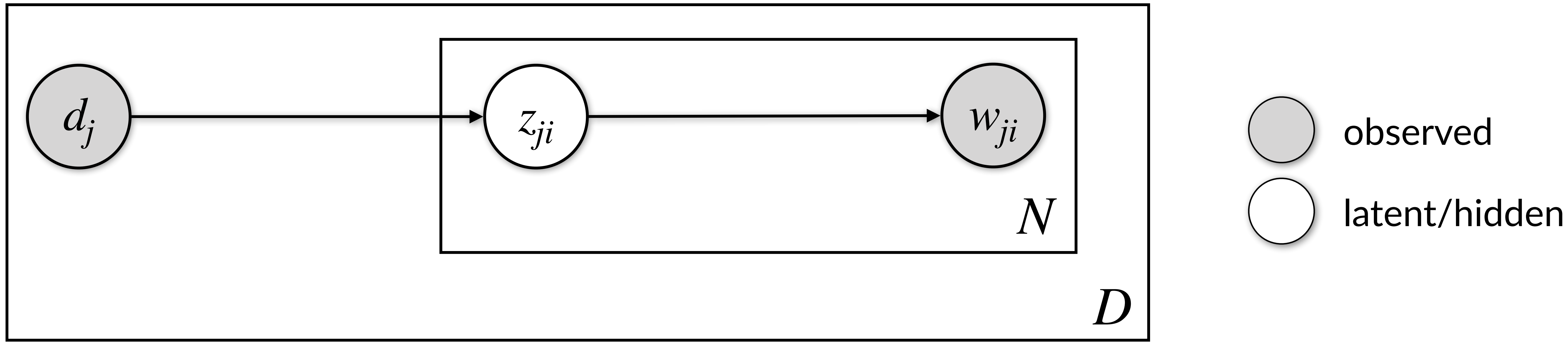
Joint probability distribution for d_j and w_i (*single word in the document*)

$$p(d_j, \mathbf{w}_j) = p(d_j) \prod_{i=1}^{N} \sum_{k=1}^{K} p(z_i = k \,|\, d_j)\, p(w_{ji} \,|\, z_i = k)$$

Joint probability distribution for d_j and $\mathbf{w}_j$ (*all words in the document*)

Probabilistic LSA — pLSA

Plate notation



Assumptions: In a document d_j, a word w_{ji} is generated from a single topic z_{ji} from the K assumed ones, and given that topic, the word is independent of all of the other words in that document.

$$p(d_j, w_i) = p(d_j)\, p(w_i \,|\, d_j) = p(d_j) \sum_{k=1}^{K} p(z = k \,|\, d_j)\, p(w_i \,|\, z = k)$$

Joint probability distribution for d_j and w_i (*single word in the document*)

$$p(d_j, \mathbf{w}_j) = p(d_j) \prod_{i=1}^{N} \sum_{k=1}^{K} p(z_i = k \,|\, d_j)\, p(w_{ji} \,|\, z_i = k)$$

Joint probability distribution for d_j and $\mathbf{w}_j$ (*all words in the document*)

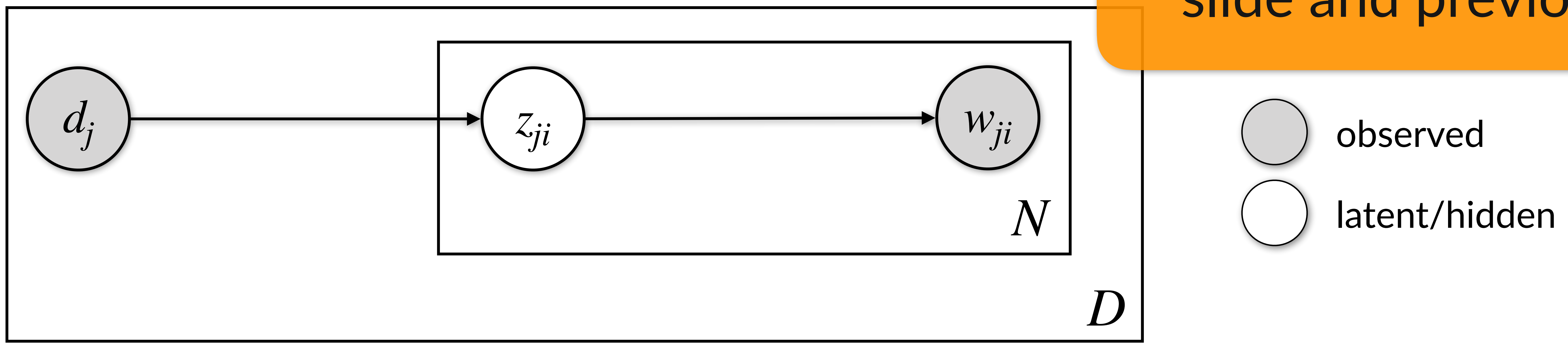
$$p(\mathbf{d}, \mathbf{W}) = \prod_{j=1}^{D} p(d_j) \prod_{i=1}^{N} \sum_{k=1}^{K} p(z_{ji} = k \,|\, d_j)\, p(w_{ji} \,|\, z_{ji} = k)$$

Joint probability distribution

Probabilistic LSA — pLSA

Find a *minor* **mistake** in this slide and previous ones

Plate notation



Assumptions: In a document d_j, a word w_{ji} is generated from a single topic z_{ji} from the K assumed ones, and given that topic, the word is independent of all of the other words in that document.

$$p(d_j, w_i) = p(d_j)\, p(w_i \,|\, d_j) = p(d_j) \sum_{k=1}^{K} p(z = k \,|\, d_j)\, p(w_i \,|\, z = k)$$

Joint probability distribution for d_j and w_i (*single word in the document*)

$$p(d_j, \mathbf{w}_j) = p(d_j) \prod_{i=1}^{N} \sum_{k=1}^{K} p(z_i = k \,|\, d_j)\, p(w_{ji} \,|\, z_i = k)$$

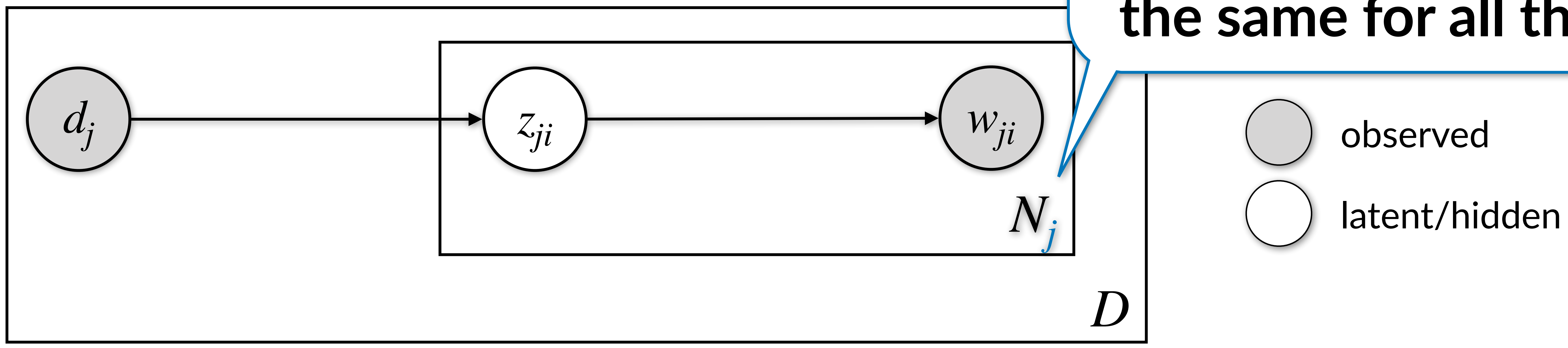
Joint probability distribution for d_j and $\mathbf{w}_j$ (*all words in the document*)

$$p(\mathbf{d}, \mathbf{W}) = \prod_{j=1}^{D} p(d_j) \prod_{i=1}^{N} \sum_{k=1}^{K} p(z_{ji} = k \,|\, d_j)\, p(w_{ji} \,|\, z_{ji} = k)$$

Joint probability distribution

Probabilistic LSA — pLSA

Plate notation



The number of words may not be the same for all the documents!

<u>**Assumptions:**</u> In a document d_j, a word w_{ji} is generated from a single topic z_{ji} from the K assumed ones, and given that topic, the word is independent of all of the other words in that document.

$$p(d_j, w_i) = p(d_j)\, p(w_i \,|\, d_j) = p(d_j) \sum_{k=1}^{K} p(z = k \,|\, d_j)\, p(w_i \,|\, z = k)$$

Joint probability distribution for d_j and w_i (*single word in the document*)

$$p(d_j, \mathbf{w}_j) = p(d_j) \prod_{i=1}^{\boxed{N_j}} \sum_{k=1}^{K} p(z_i = k \,|\, d_j)\, p(w_{ji} \,|\, z_i = k)$$

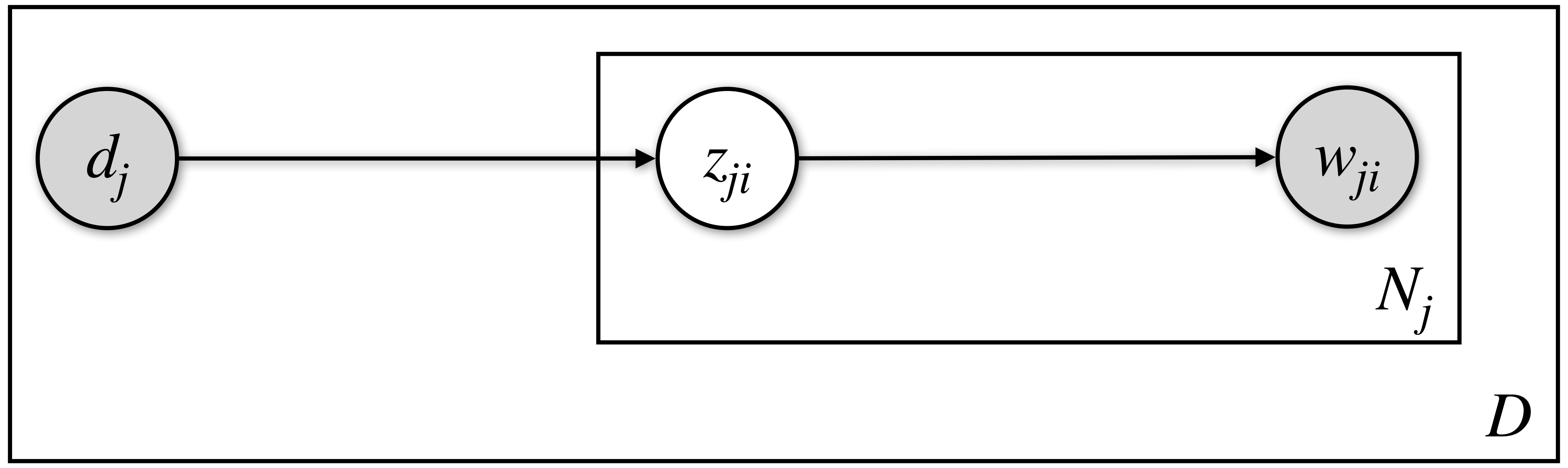
Joint probability distribution for d_j and $\mathbf{w}_j$ (*all words in the document*)

$$p(\mathbf{d}, \mathbf{W}) = \prod_{j=1}^{D} p(d_j) \prod_{i=1}^{\boxed{N_j}} \sum_{k=1}^{K} p(z_{ji} = k \,|\, d_j)\, p(w_{ji} \,|\, z_{ji} = k)$$

Joint probability distribution

pLSA — Inference

$$p(\mathbf{d}, \mathbf{W}) = \prod_{j=1}^{D} p(d_j) \prod_{i=1}^{N_j} \sum_{k=1}^{K} p(z_{ji} = k \,|\, d_j)\, p(w_{ji} \,|\, z_{ji} = k)$$

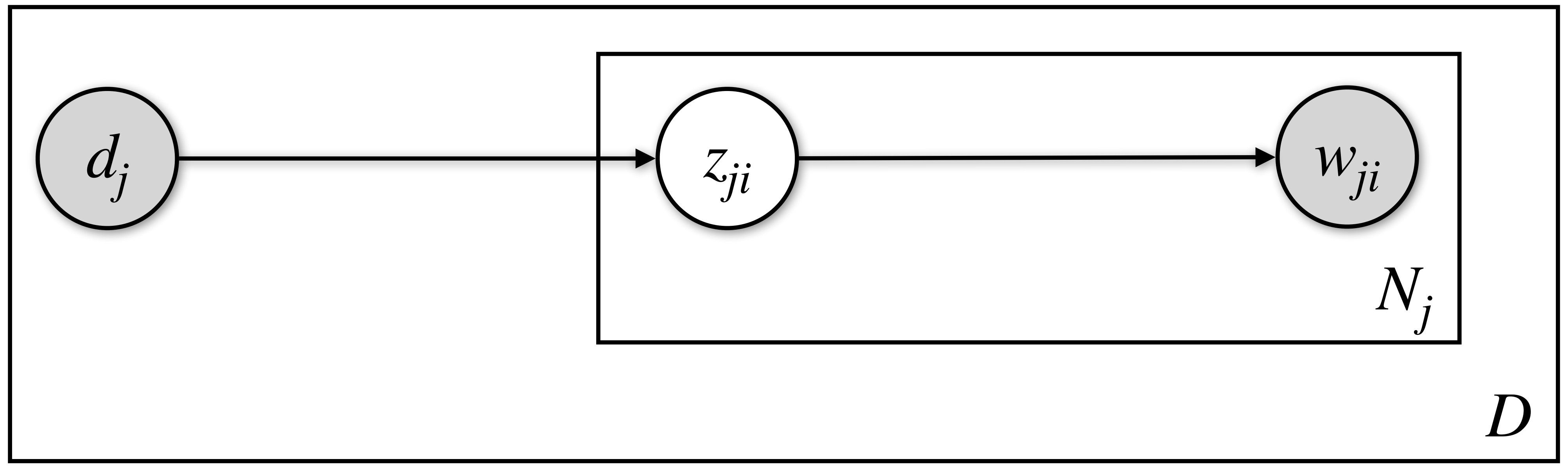


Expectation Maximisation (EM)

- **E-step:** Compute expected values of the variables, given the current parametrisation of the model. In the very beginning, start with a *random* or *uniform* parametrisation
- **M-step:** Then, using the above values, update the model parameters
- Go back to the E-step; **repeat until convergence**

pLSA — Inference

$$p(\mathbf{d}, \mathbf{W}) = \prod_{j=1}^{D} p(d_j) \prod_{i=1}^{N_j} \sum_{k=1}^{K} p(z_{ji} = k \,|\, d_j)\, p(w_{ji} \,|\, z_{ji} = k)$$

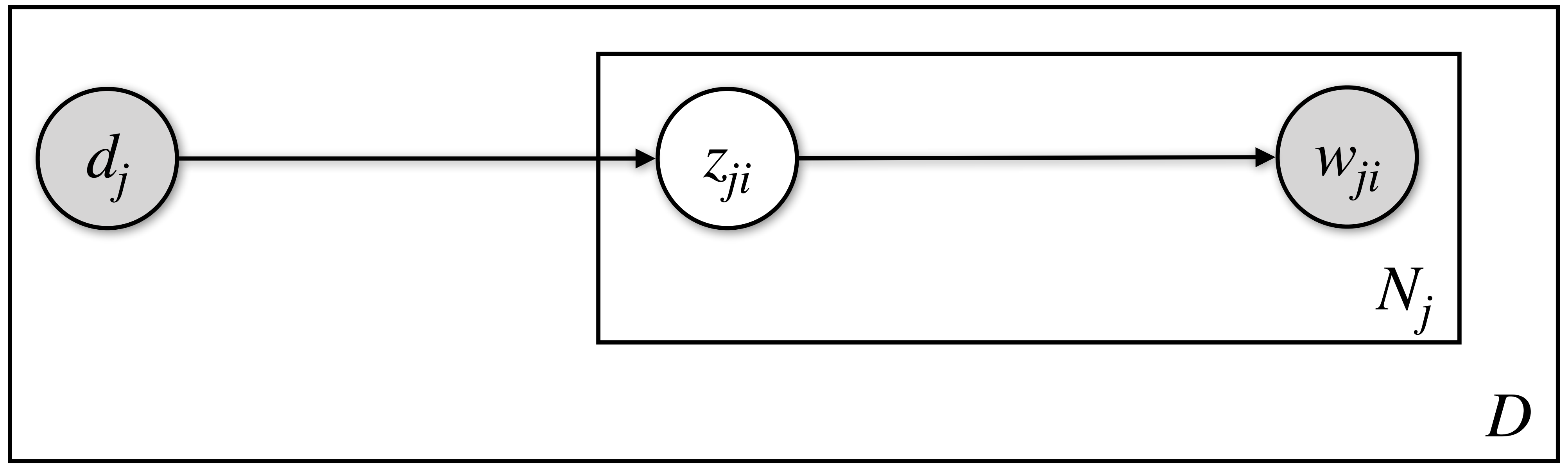


- Initialise $p(z_k \,|\, d_j)$ and $p(w_i \,|\, z_k)$ to positive quantities
- **E-step:** Estimate the probability of each topic given the words in each document

$$p(z_k \,|\, d_j, w_i) = \frac{p(z_k \,|\, d_j)\, p(w_i \,|\, z_k)}{\sum_{k'=1}^{K} p(z_{k'} \,|\, d_j)\, p(w_i \,|\, z_{k'})}$$

pLSA — Inference

$$p(\mathbf{d}, \mathbf{W}) = \prod_{j=1}^{D} p(d_j) \prod_{i=1}^{N_j} \sum_{k=1}^{K} p(z_{ji} = k \,|\, d_j)\, p(w_{ji} \,|\, z_{ji} = k)$$



- Initialise $p(z_k \,|\, d_j)$ and $p(w_i \,|\, z_k)$ to positive quantities

- **E-step:** Estimate the probability of each topic given the words in each document

$$p(z_k \,|\, d_j, w_i) = \frac{p(z_k \,|\, d_j)\, p(w_i \,|\, z_k)}{\sum_{k'=1}^{K} p(z_{k'} \,|\, d_j)\, p(w_i \,|\, z_{k'})}$$

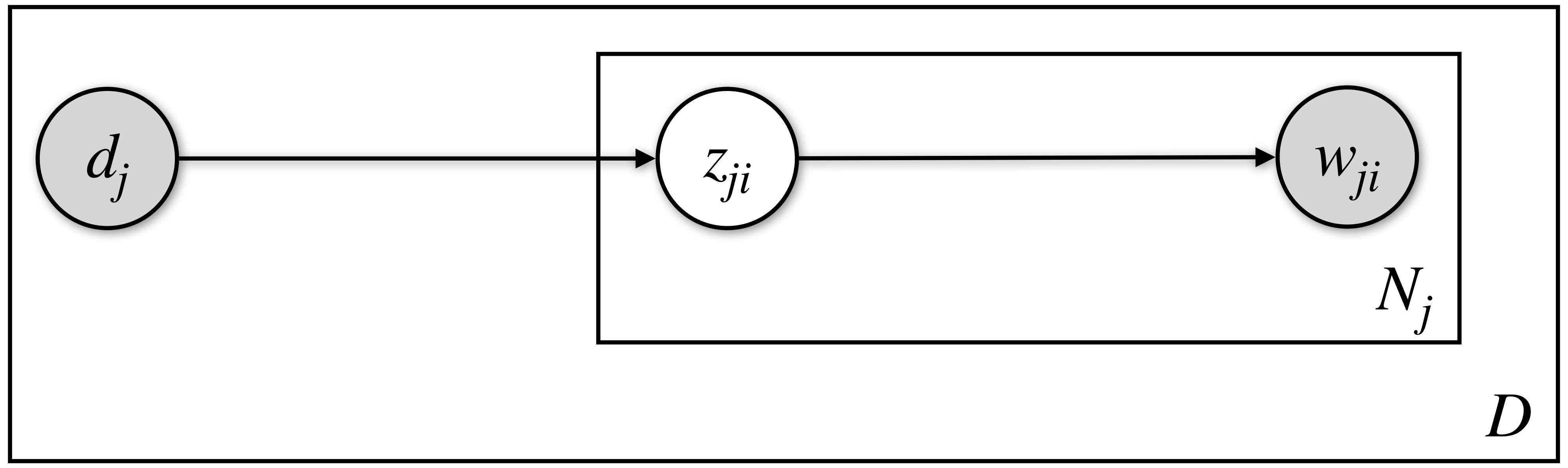
- **M-step:** Re-estimate $p(z_k \,|\, d_j)$, $p(w_i \,|\, z_k)$ given the revised $p(z_k \,|\, d_j, w_i)$

$$p(w_i \,|\, z_k) = \frac{\sum_{j=1}^{D} n(d_j, w_i)\, p(z_k \,|\, d_j, w_i)}{\sum_{j=1}^{D} \sum_{i'=1}^{N_j} n(d_j, w_{i'})\, p(z_k \,|\, d_j, w_{i'})}$$

$$p(z_k \,|\, d_j) = \frac{\sum_{i=1}^{N_j} n(d_j, w_i)\, p(z_k \,|\, d_j, w_i)}{\sum_{i=1}^{N_j} \sum_{k'=1}^{K} n(d_j, w_i)\, p(z_{k'} \,|\, d_j, w_i)}$$

pLSA — Inference

$$p(\mathbf{d}, \mathbf{W}) = \prod_{j=1}^{D} p(d_j) \prod_{i=1}^{N_j} \sum_{k=1}^{K} p(z_{ji} = k \mid d_j)\, p(w_{ji} \mid z_{ji} = k)$$



- Initialise $p(z_k \mid d_j)$ and $p(w_i \mid z_k)$ to positive quantities

- **E-step:** Estimate the probability of each topic given the words in each document

$$p(z_k \mid d_j, w_i) = \frac{p(z_k \mid d_j)\, p(w_i \mid z_k)}{\sum_{k'=1}^{K} p(z_{k'} \mid d_j)\, p(w_i \mid z_{k'})}$$

- **M-step:** Re-estimate $p(z_k \mid d_j)$, $p(w_i \mid z_k)$ given the revised $p(z_k \mid d_j, w_i)$

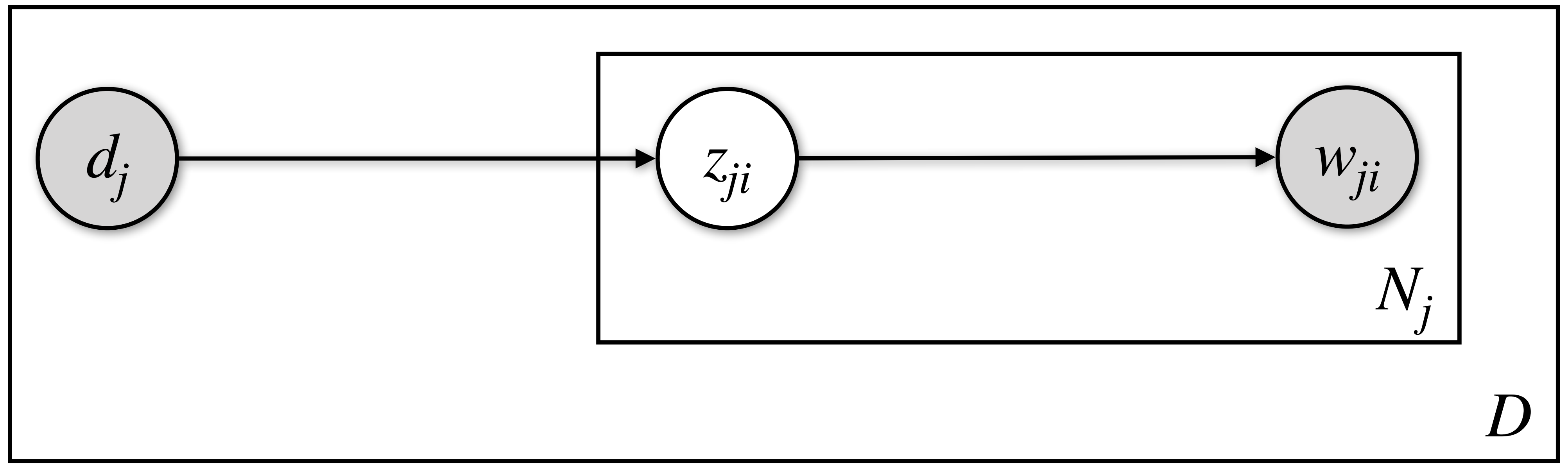
Weighted sums, e.g. $n(d_j, w_i)$ is the number of times word i appears in document j.

$$p(w_i \mid z_k) = \frac{\boxed{\sum_{j=1}^{D} n(d_j, w_i)\, p(z_k \mid d_j, w_i)}}{\sum_{j=1}^{D} \sum_{i'=1}^{N_j} n(d_j, w_{i'})\, p(z_k \mid d_j, w_{i'})}$$

$$p(z_k \mid d_j) = \frac{\boxed{\sum_{i=1}^{N_j} n(d_j, w_i)\, p(z_k \mid d_j, w_i)}}{\sum_{i=1}^{N_j} \sum_{k'=1}^{K} n(d_j, w_i)\, p(z_{k'} \mid d_j, w_i)}$$

pLSA — Inference

$$p(\mathbf{d}, \mathbf{W}) = \prod_{j=1}^{D} p(d_j) \prod_{i=1}^{N_j} \sum_{k=1}^{K} p(z_{ji} = k \,|\, d_j)\, p(w_{ji} \,|\, z_{ji} = k)$$



- Initialise $p(z_k \,|\, d_j)$ and $p(w_i \,|\, z_k)$ to positive quantities

- **E-step:** Estimate the probability of each topic given the words in each document

$$p(z_k \,|\, d_j, w_i) = \frac{p(z_k \,|\, d_j)\, p(w_i \,|\, z_k)}{\sum_{k'=1}^{K} p(z_{k'} \,|\, d_j)\, p(w_i \,|\, z_{k'})}$$

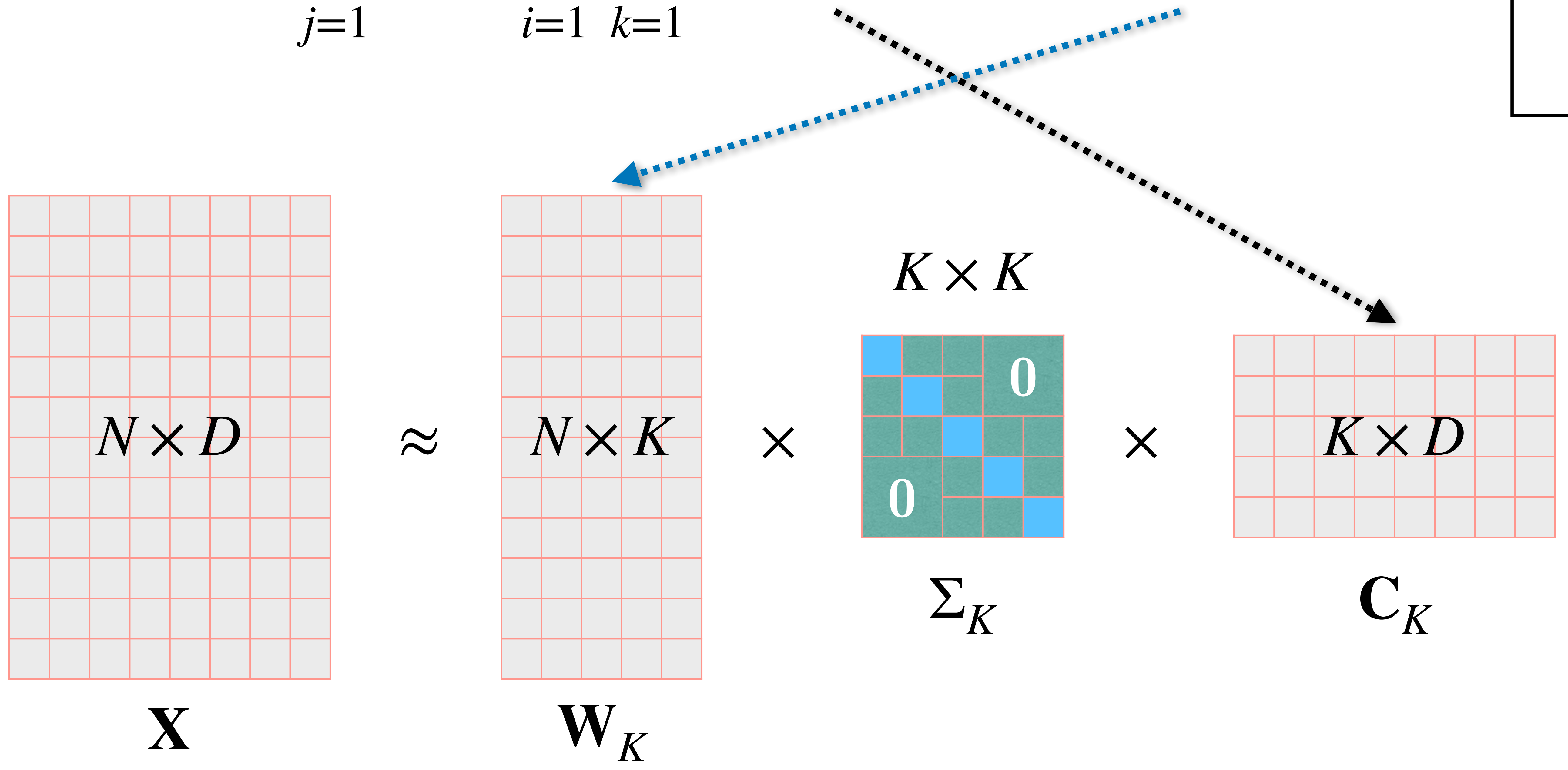
- **M-step:** Re-estimate $p(z_k \,|\, d_j)$, $p(w_i \,|\, z_k)$ given the revised $p(z_k \,|\, d_j, w_i)$

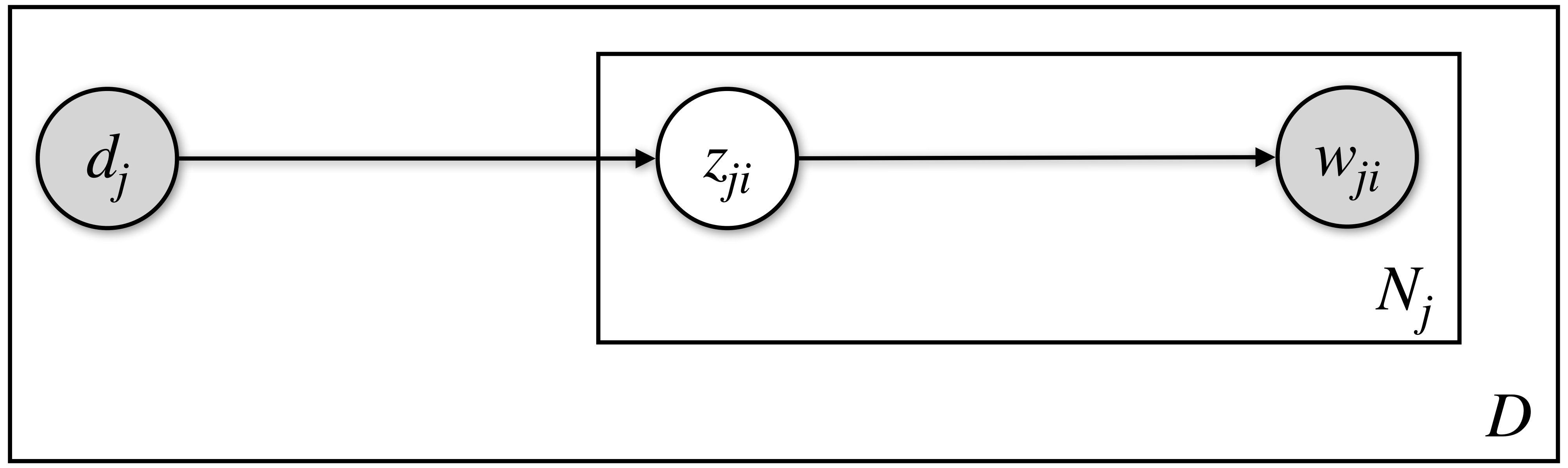
$$p(w_i \,|\, z_k) = \frac{\sum_{j=1}^{D} n(d_j, w_i)\, p(z_k \,|\, d_j, w_i)}{\sum_{j=1}^{D} \sum_{i'=1}^{N_j} n(d_j, w_{i'})\, p(z_k \,|\, d_j, w_{i'})}$$

$$p(z_k \,|\, d_j) = \frac{\sum_{i=1}^{N_j} n(d_j, w_i)\, p(z_k \,|\, d_j, w_i)}{\sum_{i=1}^{N_j} \sum_{k'=1}^{K} n(d_j, w_i)\, p(z_{k'} \,|\, d_j, w_i)}$$

pLSA and LSA

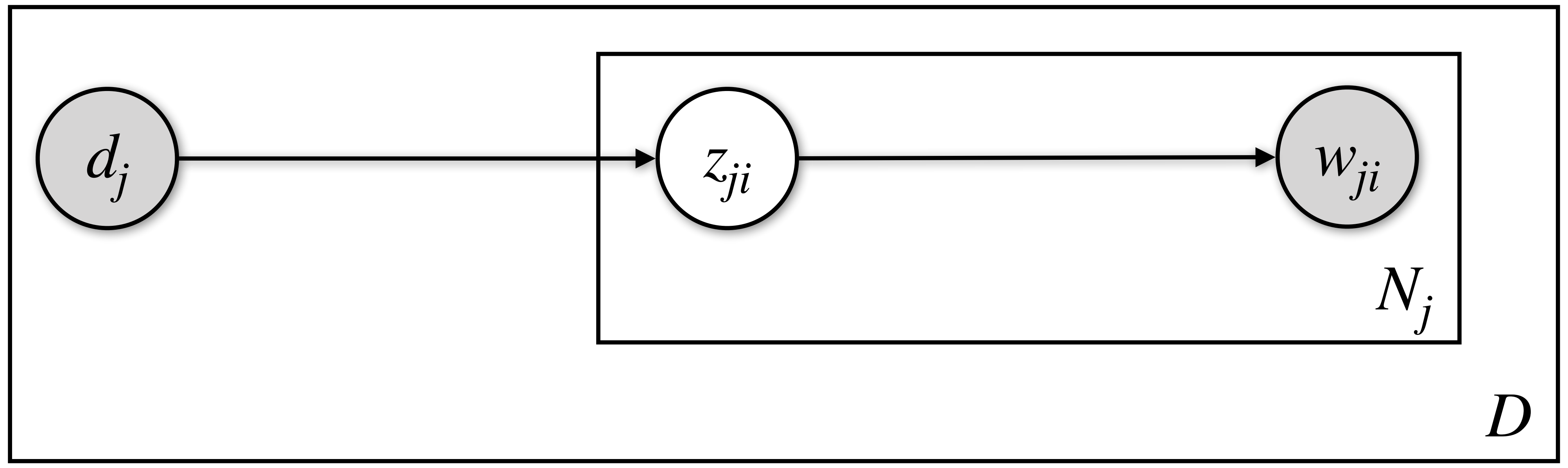
$$p(\mathbf{d},\mathbf{W}) = \prod_{j=1}^{D} p(d_j) \prod_{i=1}^{N_j} \sum_{k=1}^{K} \underline{p(z_{ji}=k\,|\,d_j)}\; \underline{p(w_{ji}\,|\,z_{ji}=k)}$$

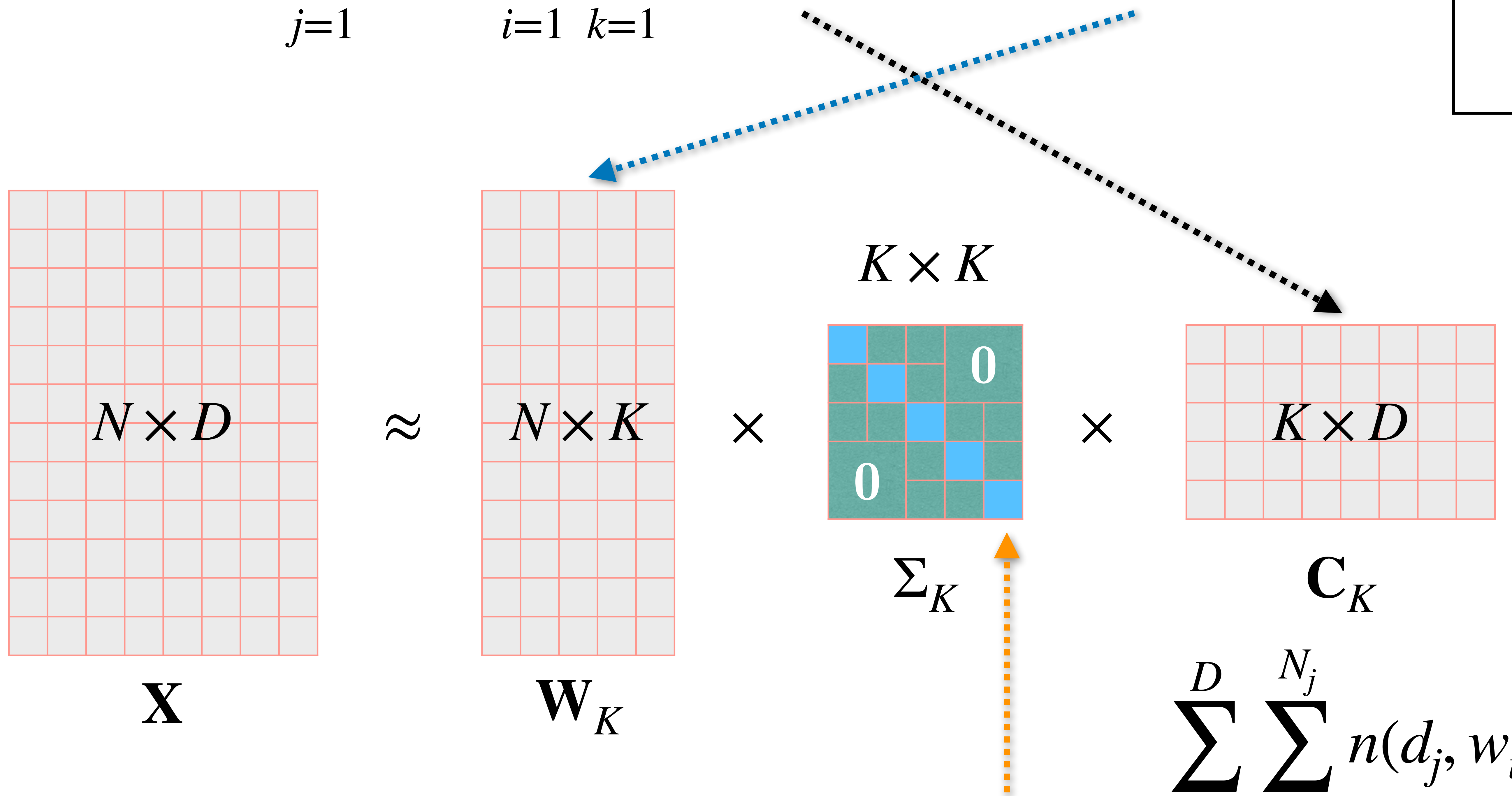




pLSA and LSA

$$p(\mathbf{d},\mathbf{W}) = \prod_{j=1}^{D} p(d_j) \prod_{i=1}^{N_j} \sum_{k=1}^{K} \underline{p(z_{ji}=k\,|\,d_j)}\; \underline{p(w_{ji}\,|\,z_{ji}=k)}$$

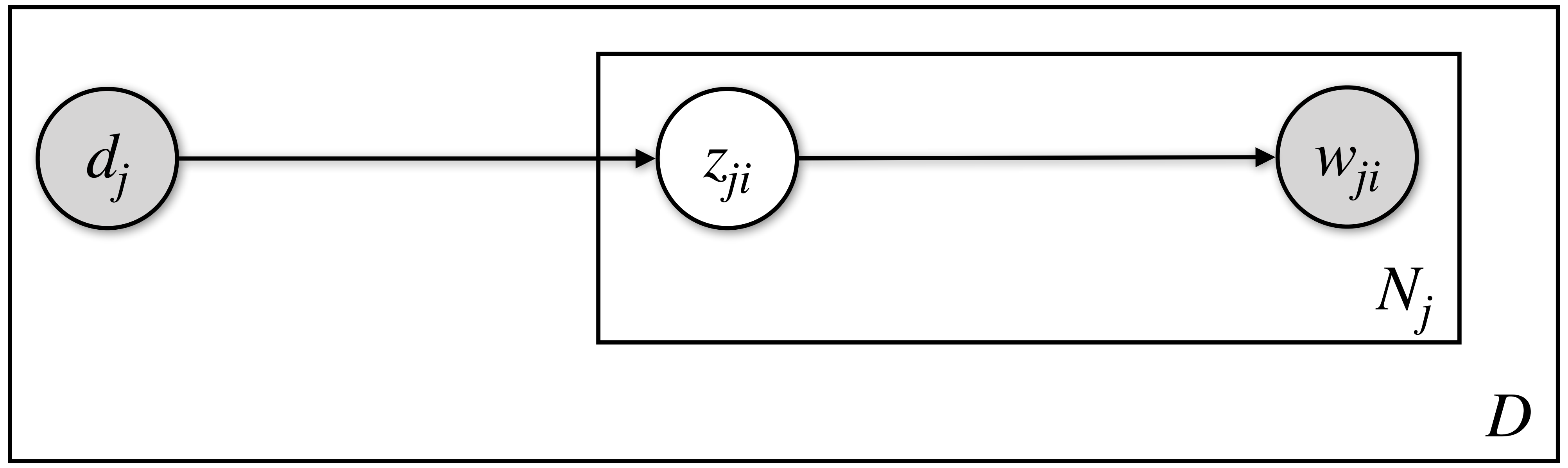


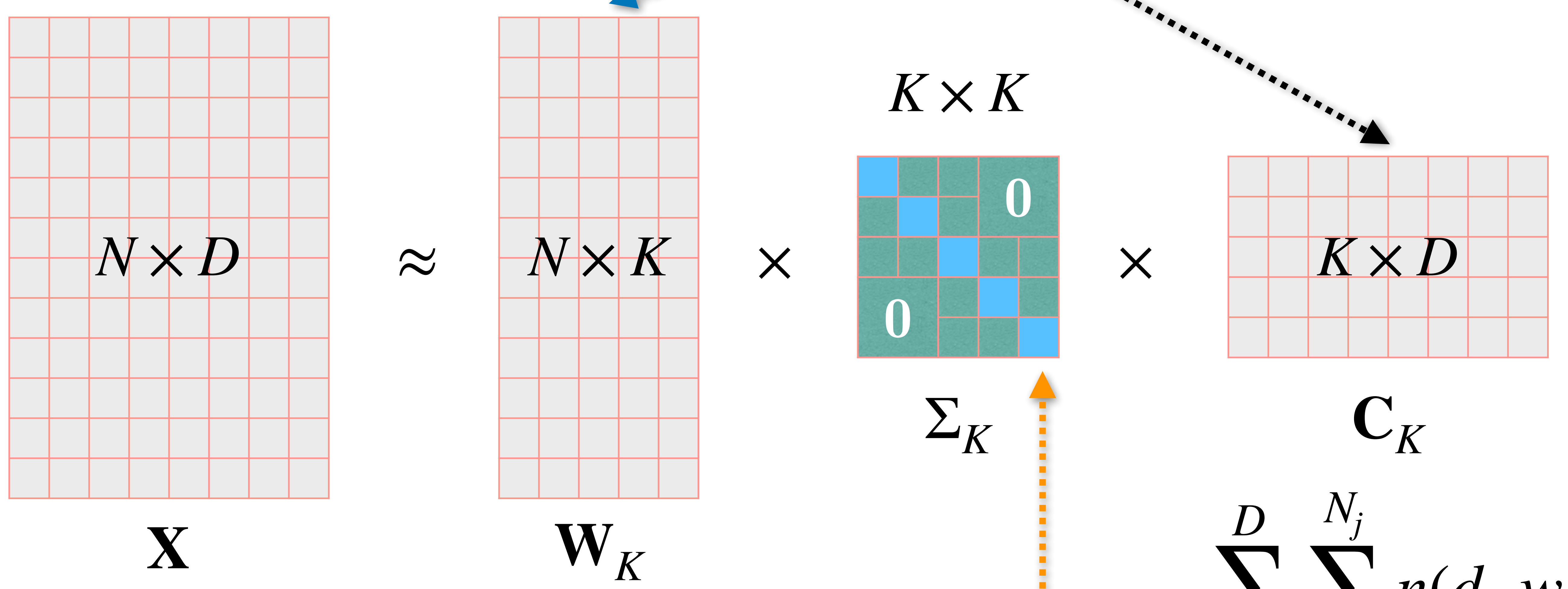


$$p(z_k) = \frac{\sum_{j=1}^{D} \sum_{i=1}^{N_j} n(d_j, w_i)\, p(z_k\,|\,d_j, w_i)}{\sum_{j=1}^{D} \sum_{i=1}^{N_j} n(d_j, w_i)}$$

pLSA and LSA

$$p(\mathbf{d}, \mathbf{W}) = \prod_{j=1}^{D} p(d_j) \prod_{i=1}^{N_j} \sum_{k=1}^{K} \underline{p(z_{ji} = k \,|\, d_j)}\; \underline{p(w_{ji} \,|\, z_{ji} = k)}$$



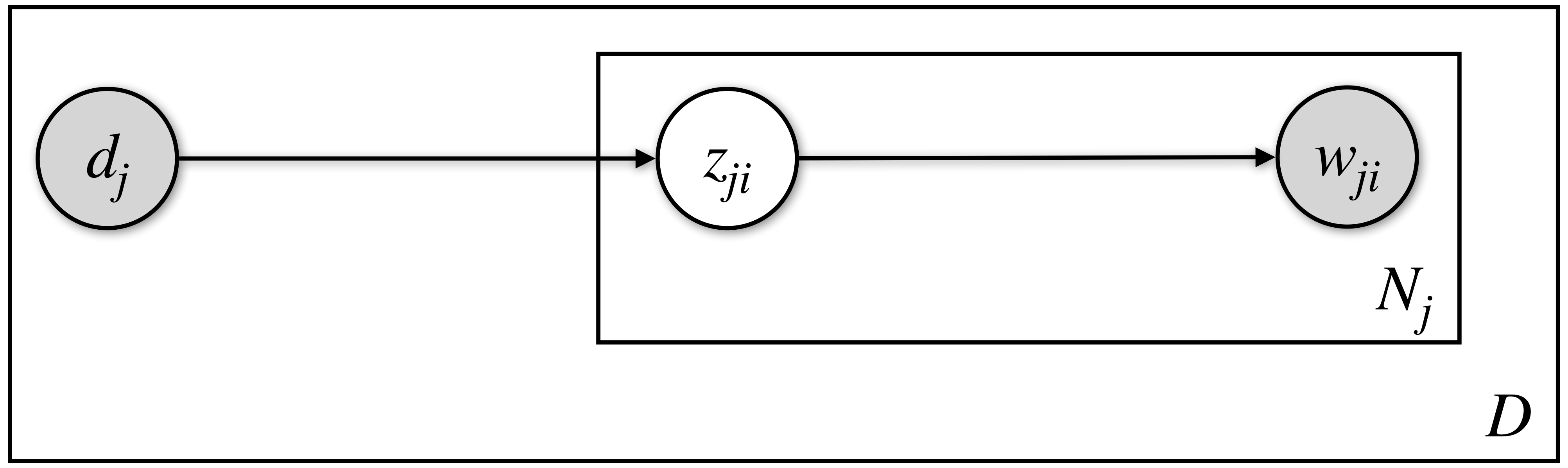


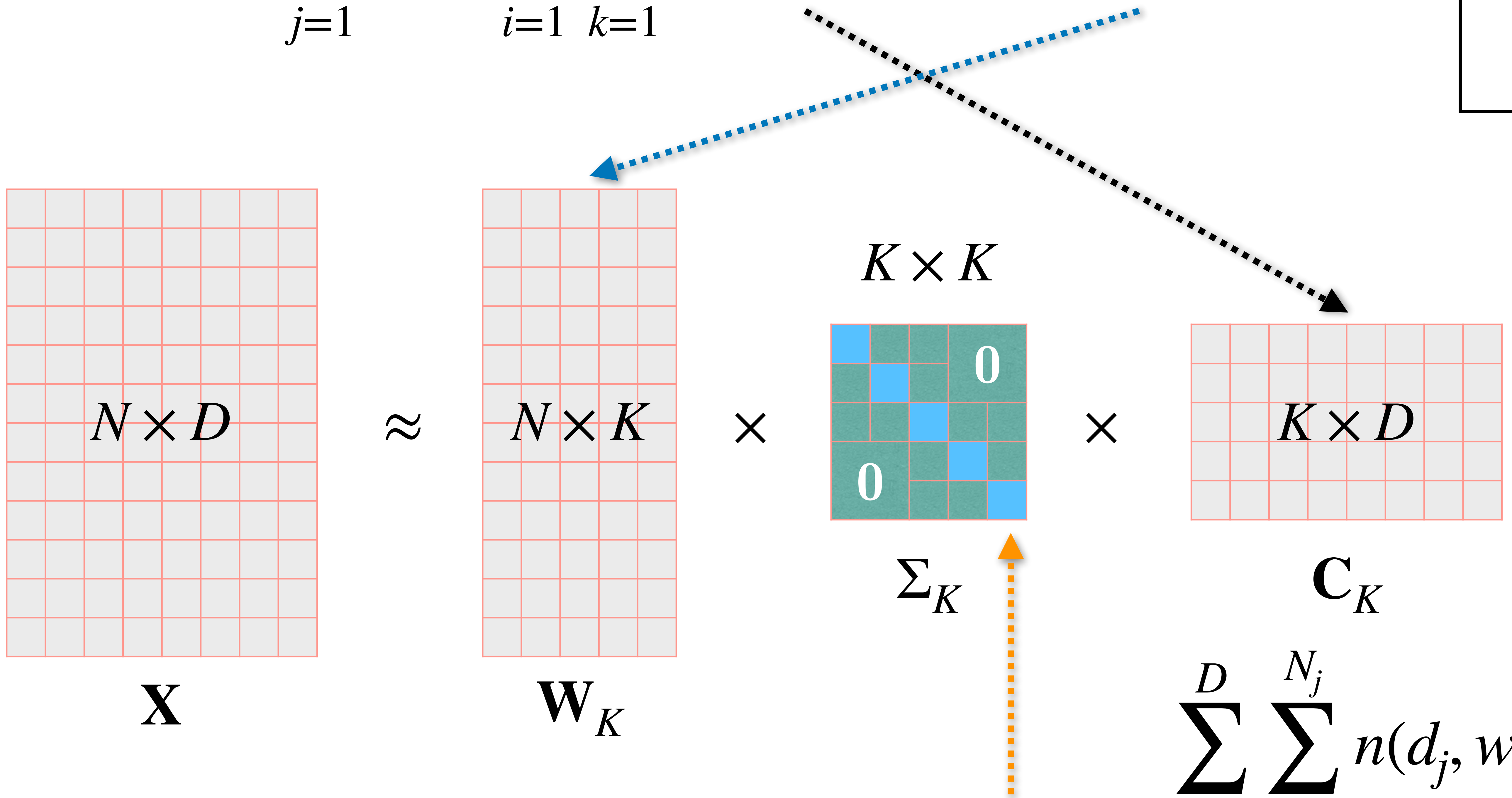
$$p(z_k) = \frac{\sum_{j=1}^{D} \sum_{i=1}^{N_j} n(d_j, w_i)\, p(z_k \,|\, d_j, w_i)}{\sum_{j=1}^{D} \sum_{i=1}^{N_j} n(d_j, w_i)}$$

So, why are LSA and pLSA different?

pLSA and LSA

$$p(\mathbf{d}, \mathbf{W}) = \prod_{j=1}^{D} p(d_j) \prod_{i=1}^{N_j} \sum_{k=1}^{K} \underline{p(z_{ji} = k \,|\, d_j)}\; \underline{p(w_{ji} \,|\, z_{ji} = k)}$$





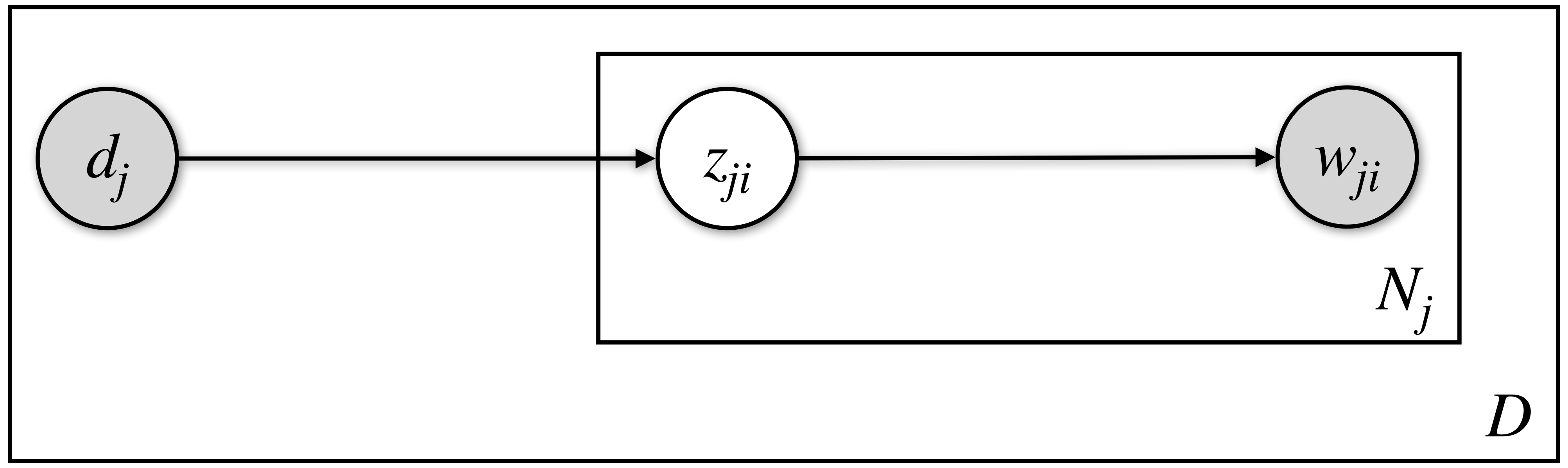
Main difference

The two techniques have a different objective function — probabilistic vs. deterministic approach

$$p(z_k) = \frac{\sum_{j=1}^{D} \sum_{i=1}^{N_j} n(d_j, w_i)\, p(z_k \,|\, d_j, w_i)}{\sum_{j=1}^{D} \sum_{i=1}^{N_j} n(d_j, w_i)}$$

pLSA — Disadvantages

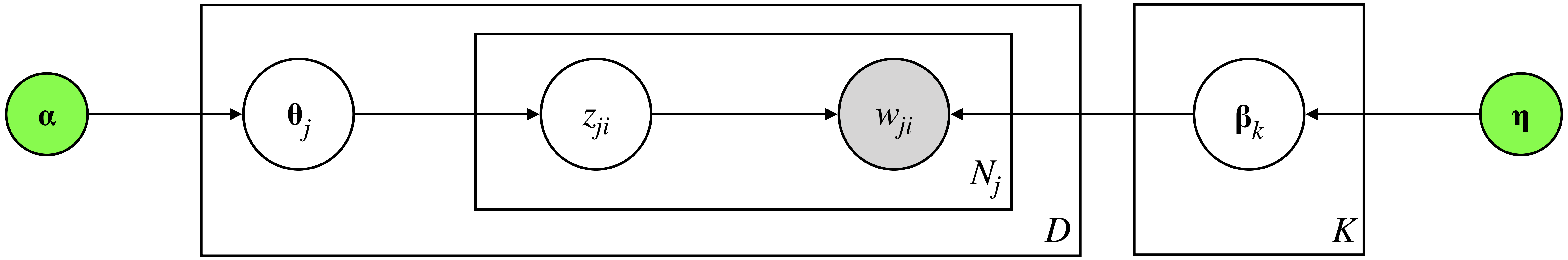
$$p(\mathbf{d}, \mathbf{W}) = \prod_{j=1}^{D} p(d_j) \prod_{i=1}^{N_j} \sum_{k=1}^{K} p(z_{ji} = k \,|\, d_j)\, p(w_{ji} \,|\, z_{ji} = k)$$



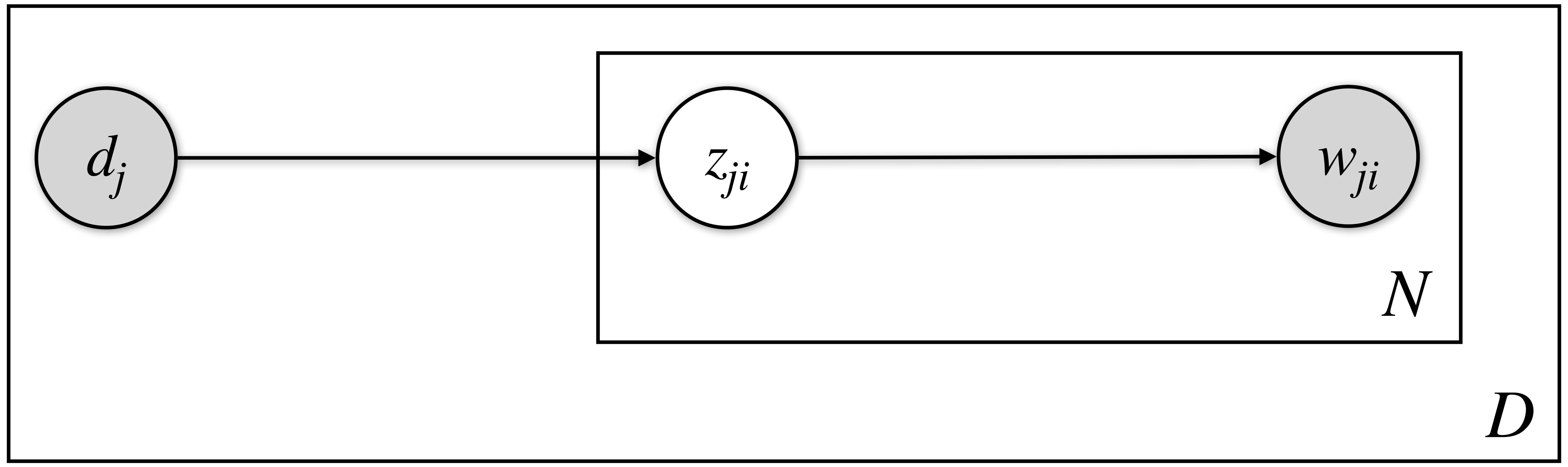
- The number of parameters that we need to learn during training grows linearly with the number of documents (D), which ultimately leads to overfitting.

- pLSA learns $p(z_k \,|\, d_j)$ only for the documents it sees during the training phase. To deal with a new document, it needs to repeat EM (retrain). *Not the best thing to do for large and live document collections!*

Latent Dirichlet Allocation (LDA)

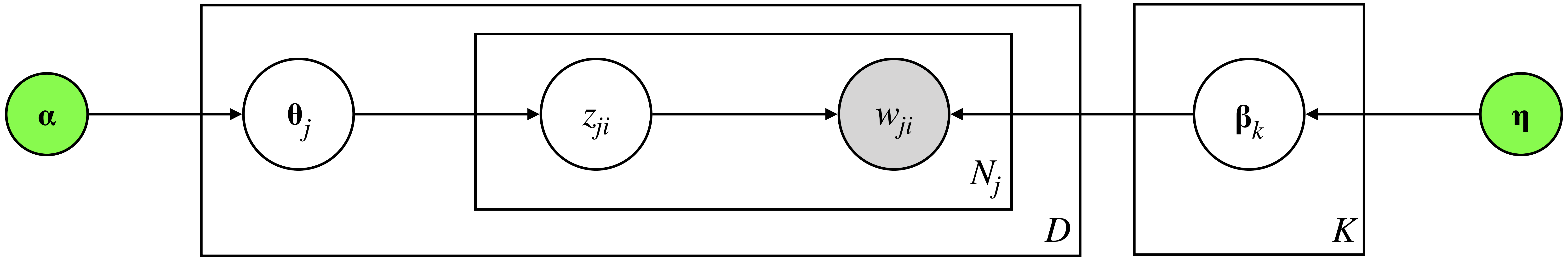


LDA



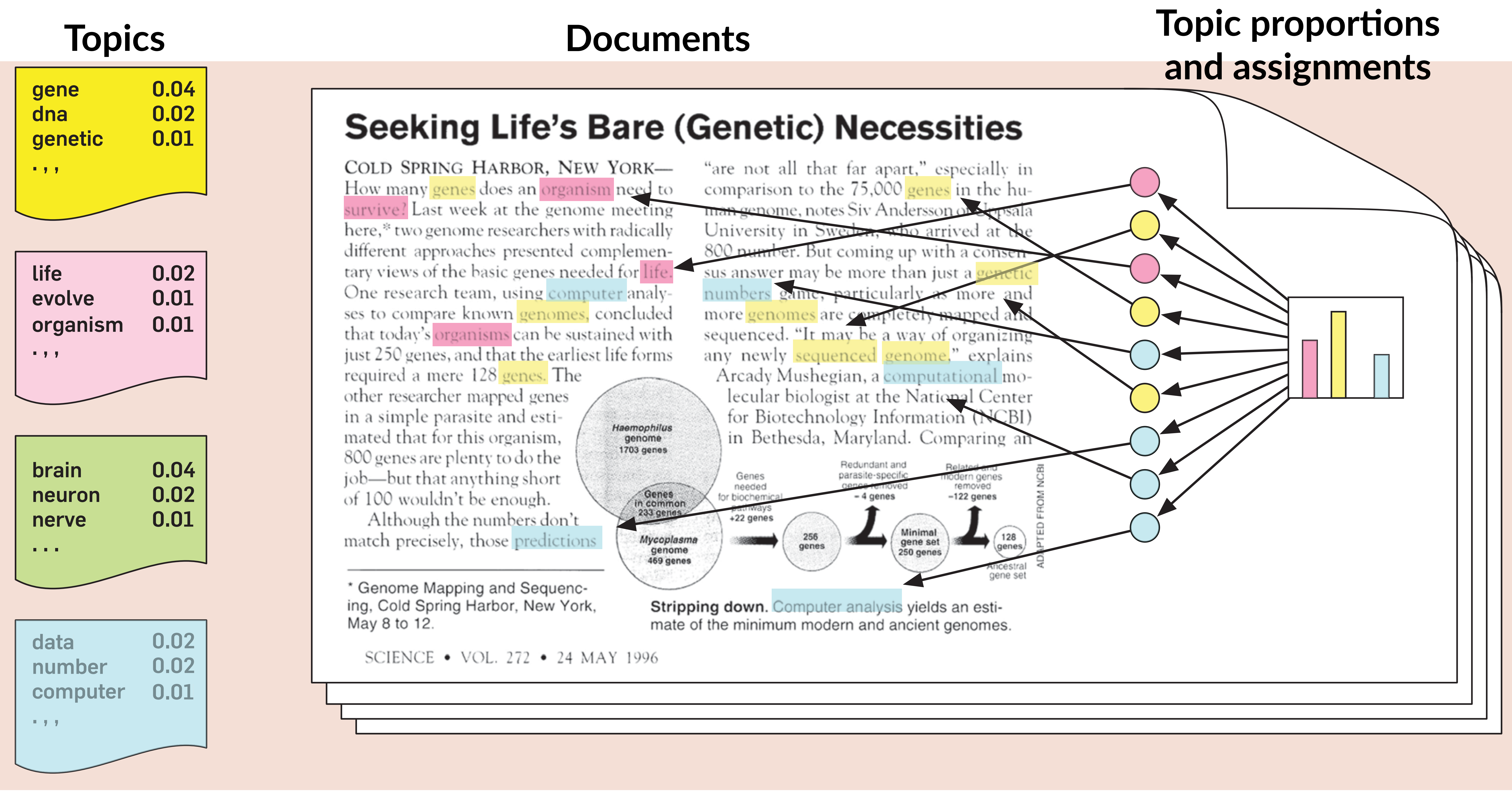
pLSA

Latent Dirichlet Allocation (LDA)



1. For each of the K topics draw a multinomial distribution (over words) $\boldsymbol{\beta}_k$ from a Dirichlet distribution with parameter $\boldsymbol{\eta}$
2. For each of the D documents draw a multinomial distribution (over topics) $\boldsymbol{\theta}_j$ from a Dirichlet distribution with parameter $\boldsymbol{\alpha}$
3. For each word position i (1 to N_j) in a document j:
    a. Select a latent topic z_{ji} from the multinomial distribution (step 2) parametrised by $\boldsymbol{\theta}_j$
    b. Choose the observed word w_{ji} from the multinomial distribution (step 1) parametrised by $\boldsymbol{\beta}_{z_{ji}}$

LDA — Generative story



gene	0.04
dna	0.02
genetic	0.01
. , ,	

life	0.02
evolve	0.01
organism	0.01
. , ,	

brain	0.04
neuron	0.02
nerve	0.01
. . .	

data	0.02
number	0.02
computer	0.01
. , ,	

Seeking Life's Bare (Genetic) Necessities

COLD SPRING HARBOR, NEW YORK—How many genes does an organism need to survive? Last week at the genome meeting here,* two genome researchers with radically different approaches presented complementary views of the basic genes needed for life. One research team, using computer analyses to compare known genomes, concluded that today's organisms can be sustained with just 250 genes, and that the earliest life forms required a mere 128 genes. The other researcher mapped genes in a simple parasite and estimated that for this organism, 800 genes are plenty to do the job—but that anything short of 100 wouldn't be enough.

Although the numbers don't match precisely, those predictions "are not all that far apart," especially in comparison to the 75,000 genes in the human genome, notes Siv Andersson of Uppsala University in Sweden, who arrived at the 800 number. But coming up with a consensus answer may be more than just a genetic numbers game, particularly as more and more genomes are completely mapped and sequenced. "It may be a way of organizing any newly sequenced genome," explains Arcady Mushegian, a computational molecular biologist at the National Center for Biotechnology Information (NCBI) in Bethesda, Maryland. Comparing an

Haemophilus genome 1703 genes
Genes in common 233 genes
Mycoplasma genome 469 genes
Genes needed for biochemical pathways +22 genes
256 genes
Redundant and parasite-specific genes removed −4 genes
Minimal gene set 250 genes
Related and modern genes removed −122 genes
128 genes
Ancestral gene set
ADAPTED FROM NCBI

Stripping down. Computer analysis yields an estimate of the minimum modern and ancient genomes.

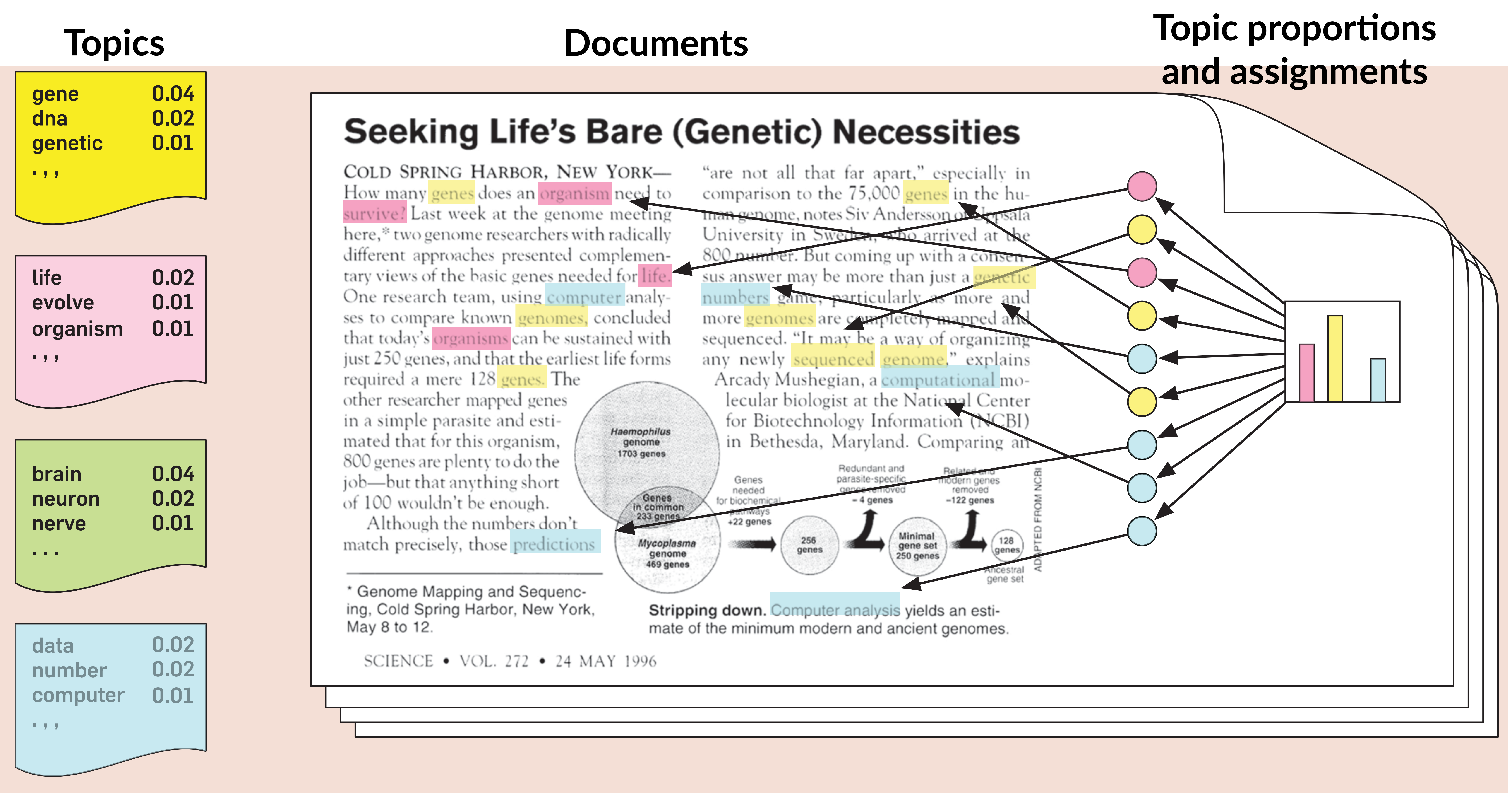
* Genome Mapping and Sequencing, Cold Spring Harbor, New York, May 8 to 12.

SCIENCE • VOL. 272 • 24 MAY 1996

Assume a number of topics, defined as distributions over words (far left). A document is generated by first choosing a distribution over the topics (far right), then for each word position choosing a topic assignment (coloured coins), then choosing a word from the corresponding topic.

Blei. CACM, 2012, doi.org/10.1145/2133806.2133826

LDA — Generative story



Topics

gene	0.04
dna	0.02
genetic	0.01
. , ,	

life	0.02
evolve	0.01
organism	0.01
. , ,	

brain	0.04
neuron	0.02
nerve	0.01
. . .	

data	0.02
number	0.02
computer	0.01
. , ,	

Documents

Topic proportions and assignments

Seeking Life's Bare (Genetic) Necessities

COLD SPRING HARBOR, NEW YORK—How many genes does an organism need to survive? Last week at the genome meeting here,* two genome researchers with radically different approaches presented complementary views of the basic genes needed for life. One research team, using computer analyses to compare known genomes, concluded that today's organisms can be sustained with just 250 genes, and that the earliest life forms required a mere 128 genes. The other researcher mapped genes in a simple parasite and estimated that for this organism, 800 genes are plenty to do the job—but that anything short of 100 wouldn't be enough.

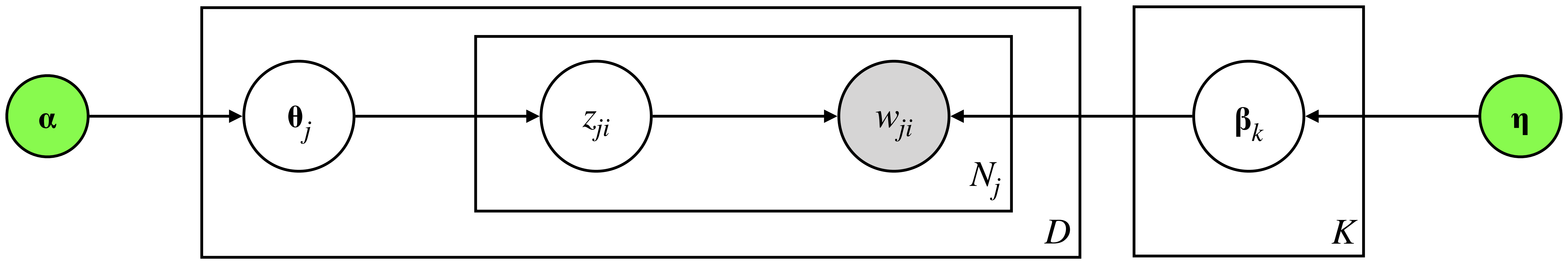
Although the numbers don't match precisely, those predictions "are not all that far apart," especially in comparison to the 75,000 genes in the human genome, notes Siv Andersson of Uppsala University in Sweden, who arrived at the 800 number. But coming up with a consensus answer may be more than just a genetic numbers game, particularly as more and more genomes are completely mapped and sequenced. "It may be a way of organizing any newly sequenced genome," explains Arcady Mushegian, a computational molecular biologist at the National Center for Biotechnology Information (NCBI) in Bethesda, Maryland. Comparing an

Haemophilus genome 1703 genes
Genes in common 233 genes
Mycoplasma genome 469 genes
Genes needed for biochemical pathways +22 genes
256 genes
Redundant and parasite-specific genes removed −4 genes
Minimal gene set 250 genes
Related and modern genes removed −122 genes
128 genes
Ancestral gene set
ADAPTED FROM NCBI

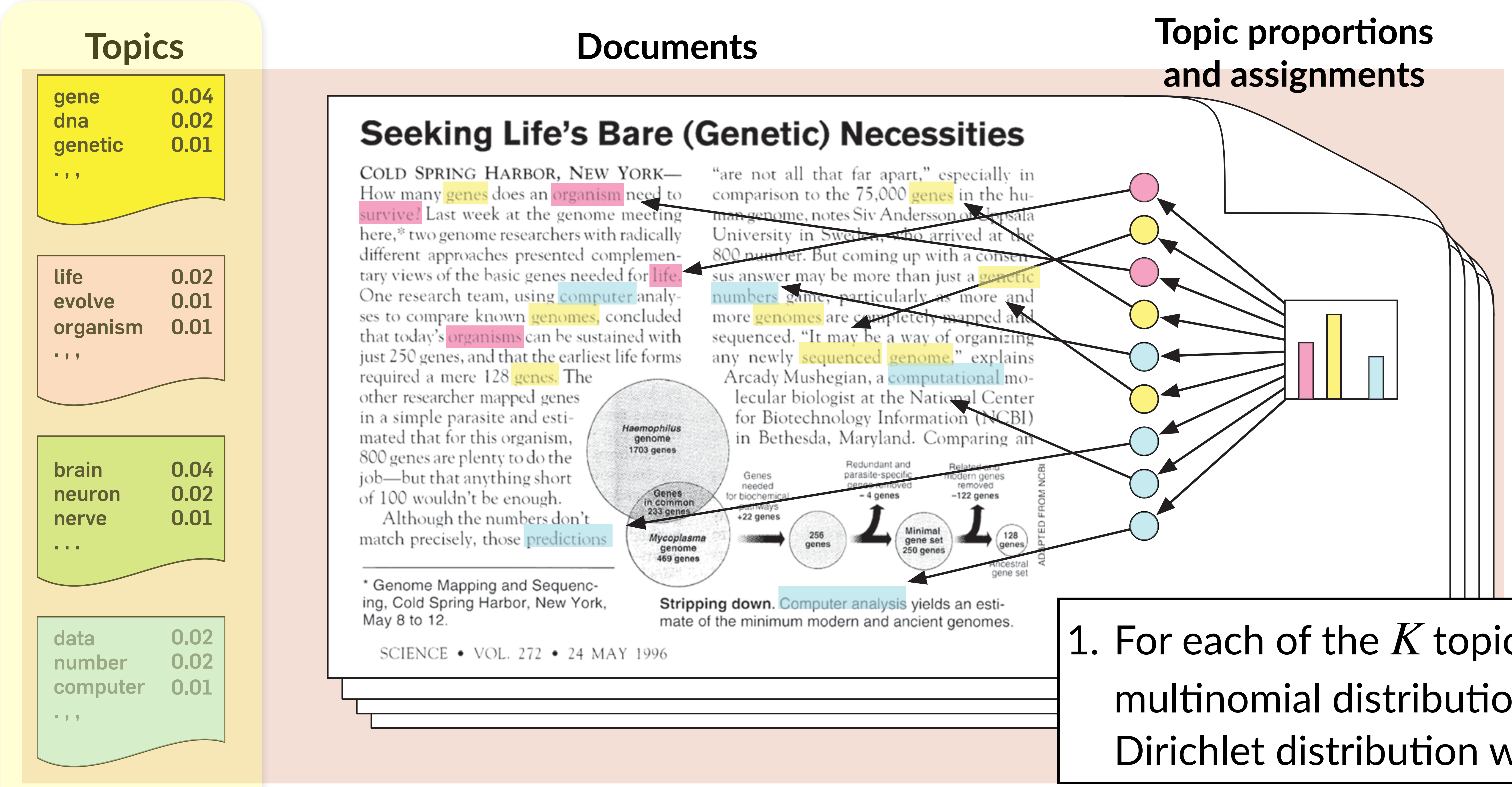
Stripping down. Computer analysis yields an estimate of the minimum modern and ancient genomes.

* Genome Mapping and Sequencing, Cold Spring Harbor, New York, May 8 to 12.

SCIENCE • VOL. 272 • 24 MAY 1996



LDA — Generative story



gene	0.04
dna	0.02
genetic	0.01
,,,	

life	0.02
evolve	0.01
organism	0.01
,,,	

brain	0.04
neuron	0.02
nerve	0.01
...	

data	0.02
number	0.02
computer	0.01
,,,	

Seeking Life's Bare (Genetic) Necessities

COLD SPRING HARBOR, NEW YORK—How many genes does an organism need to survive? Last week at the genome meeting here,* two genome researchers with radically different approaches presented complementary views of the basic genes needed for life. One research team, using computer analyses to compare known genomes, concluded that today's organisms can be sustained with just 250 genes, and that the earliest life forms required a mere 128 genes. The other researcher mapped genes in a simple parasite and estimated that for this organism, 800 genes are plenty to do the job—but that anything short of 100 wouldn't be enough.

Although the numbers don't match precisely, those predictions "are not all that far apart," especially in comparison to the 75,000 genes in the human genome, notes Siv Andersson of Uppsala University in Sweden, who arrived at the 800 number. But coming up with a consensus answer may be more than just a genetic numbers game, particularly as more and more genomes are completely mapped and sequenced. "It may be a way of organizing any newly sequenced genome," explains Arcady Mushegian, a computational molecular biologist at the National Center for Biotechnology Information (NCBI) in Bethesda, Maryland. Comparing an

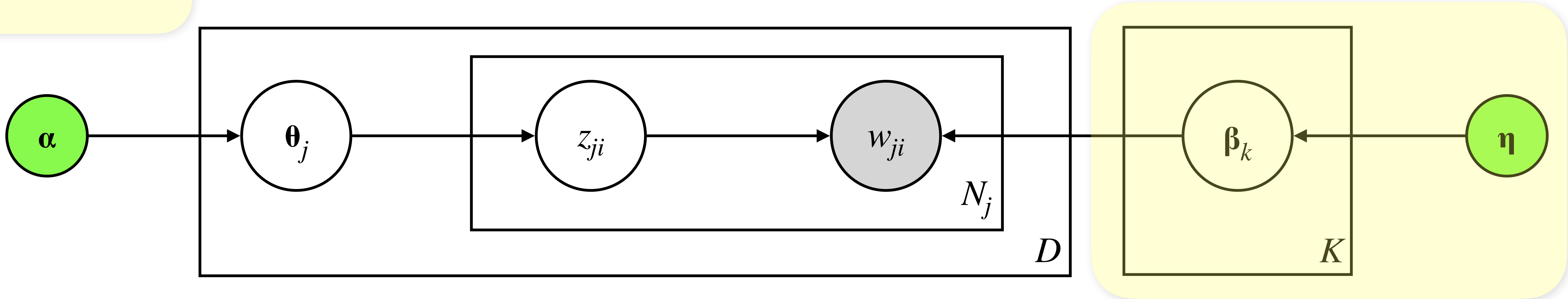
Haemophilus genome 1703 genes
Genes in common 233 genes
Mycoplasma genome 469 genes
Genes needed for biochemical pathways +22 genes
256 genes
Redundant and parasite-specific genes removed −4 genes
Minimal gene set 250 genes
Related and modern genes removed −122 genes
128 genes
Ancestral gene set
ADAPTED FROM NCBI

Stripping down. Computer analysis yields an estimate of the minimum modern and ancient genomes.

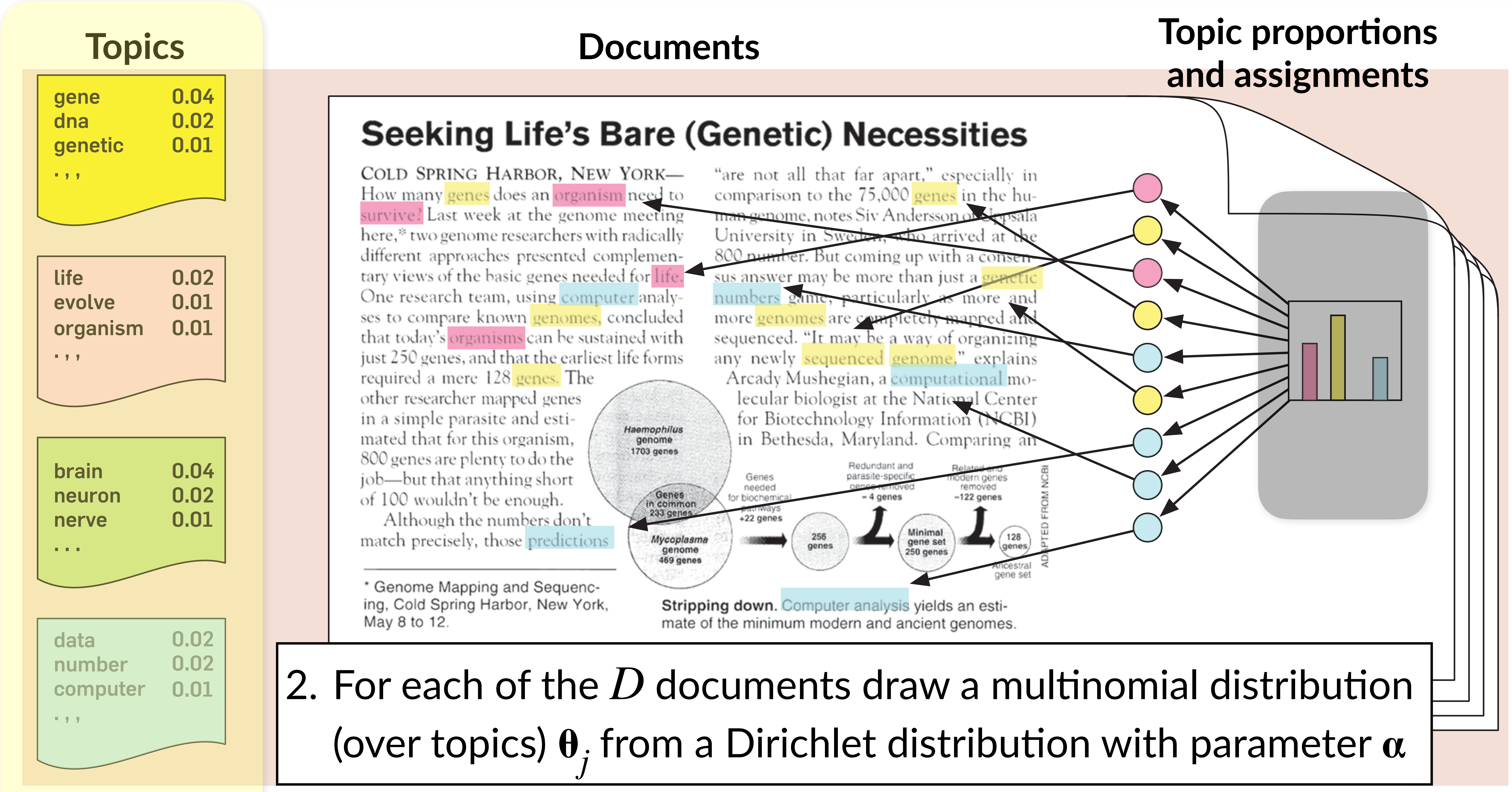
* Genome Mapping and Sequencing, Cold Spring Harbor, New York, May 8 to 12.

SCIENCE • VOL. 272 • 24 MAY 1996

1. For each of the K topics draw a multinomial distribution $\boldsymbol{\beta}_k$ from a Dirichlet distribution with parameter $\boldsymbol{\eta}$



LDA — Generative story



gene	0.04
dna	0.02
genetic	0.01
,,,	

life	0.02
evolve	0.01
organism	0.01
,,,	

brain	0.04
neuron	0.02
nerve	0.01
. . .	

data	0.02
number	0.02
computer	0.01
,,,	

Seeking Life's Bare (Genetic) Necessities

COLD SPRING HARBOR, NEW YORK— How many genes does an organism need to survive? Last week at the genome meeting here,* two genome researchers with radically different approaches presented complementary views of the basic genes needed for life. One research team, using computer analyses to compare known genomes, concluded that today's organisms can be sustained with just 250 genes, and that the earliest life forms required a mere 128 genes. The other researcher mapped genes in a simple parasite and estimated that for this organism, 800 genes are plenty to do the job—but that anything short of 100 wouldn't be enough.

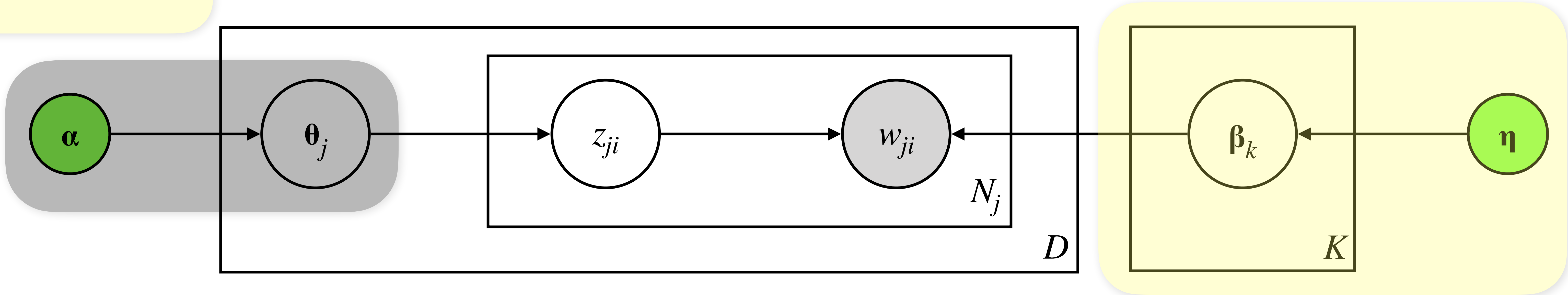
Although the numbers don't match precisely, those predictions "are not all that far apart," especially in comparison to the 75,000 genes in the human genome, notes Siv Andersson of Uppsala University in Sweden, who arrived at the 800 number. But coming up with a consensus answer may be more than just a genetic numbers game, particularly as more and more genomes are completely mapped and sequenced. "It may be a way of organizing any newly sequenced genome," explains Arcady Mushegian, a computational molecular biologist at the National Center for Biotechnology Information (NCBI) in Bethesda, Maryland. Comparing an

Haemophilus genome 1703 genes
Genes in common 233 genes
Mycoplasma genome 469 genes
Genes needed for biochemical pathways +22 genes
256 genes
Redundant and parasite-specific genes removed − 4 genes
Minimal gene set 250 genes
Related and modern genes removed −122 genes
128 genes
Ancestral gene set
ADAPTED FROM NCBI

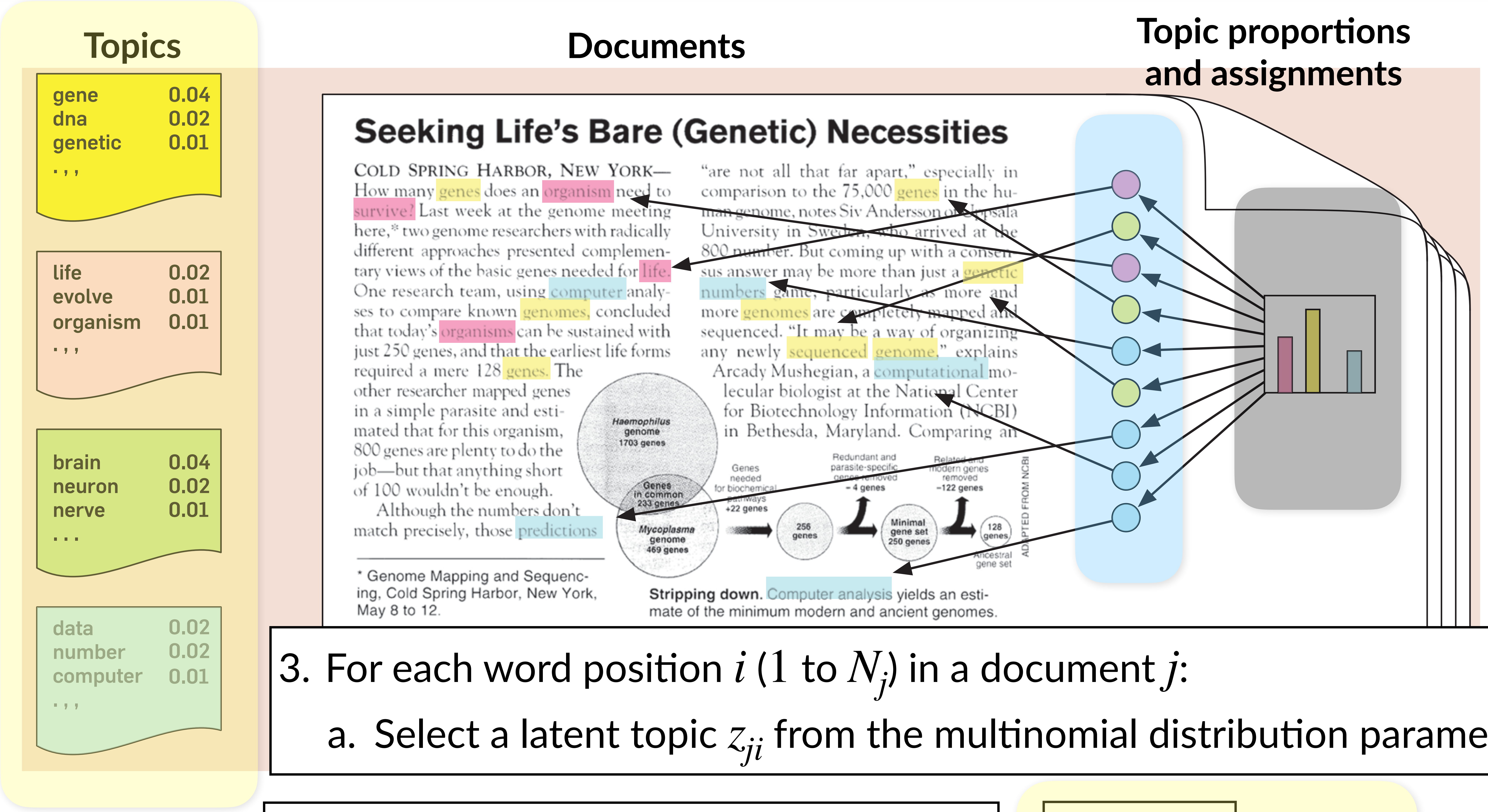
Stripping down. Computer analysis yields an estimate of the minimum modern and ancient genomes.

* Genome Mapping and Sequencing, Cold Spring Harbor, New York, May 8 to 12.

2. For each of the D documents draw a multinomial distribution (over topics) $\boldsymbol{\theta}_j$ from a Dirichlet distribution with parameter $\boldsymbol{\alpha}$



LDA — Generative story


Topics
gene 0.04
dna 0.02
genetic 0.01
,,,
life 0.02
evolve 0.01
organism 0.01
,,,
brain 0.04
neuron 0.02
nerve 0.01
...
data 0.02
number 0.02
computer 0.01
,,,
Documents
Topic proportions and assignments
Haemophilus genome 1703 genes
Genes in common 233 genes
Mycoplasma genome 469 genes
Genes needed for biochemical pathways +22 genes
256 genes
Redundant and parasite-specific genes removed −4 genes
Minimal gene set 250 genes
Related and modern genes removed −122 genes
128 genes
Ancestral gene set
ADAPTED FROM NCBI

Seeking Life's Bare (Genetic) Necessities

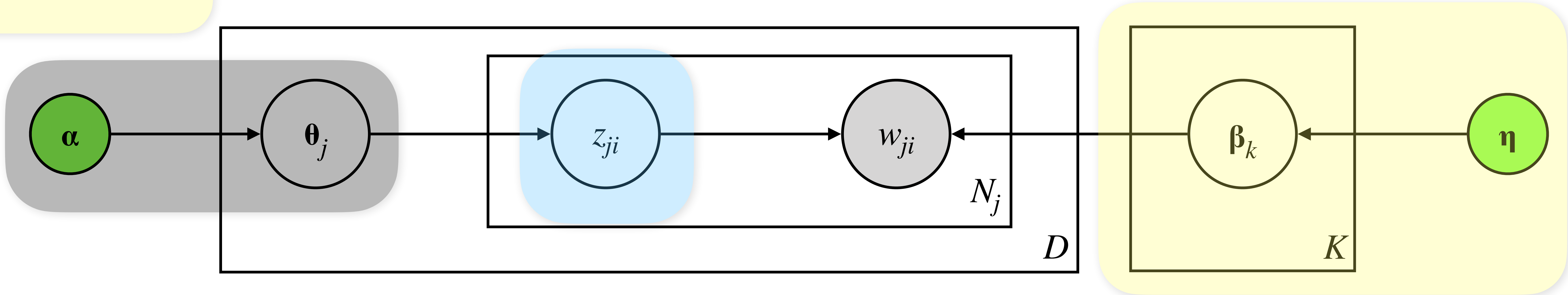
COLD SPRING HARBOR, NEW YORK— How many genes does an organism need to survive? Last week at the genome meeting here,* two genome researchers with radically different approaches presented complementary views of the basic genes needed for life. One research team, using computer analyses to compare known genomes, concluded that today's organisms can be sustained with just 250 genes, and that the earliest life forms required a mere 128 genes. The other researcher mapped genes in a simple parasite and estimated that for this organism, 800 genes are plenty to do the job—but that anything short of 100 wouldn't be enough.

Although the numbers don't match precisely, those predictions "are not all that far apart," especially in comparison to the 75,000 genes in the human genome, notes Siv Andersson of Uppsala University in Sweden, who arrived at the 800 number. But coming up with a consensus answer may be more than just a genetic numbers game, particularly as more and more genomes are completely mapped and sequenced. "It may be a way of organizing any newly sequenced genome," explains Arcady Mushegian, a computational molecular biologist at the National Center for Biotechnology Information (NCBI) in Bethesda, Maryland. Comparing an

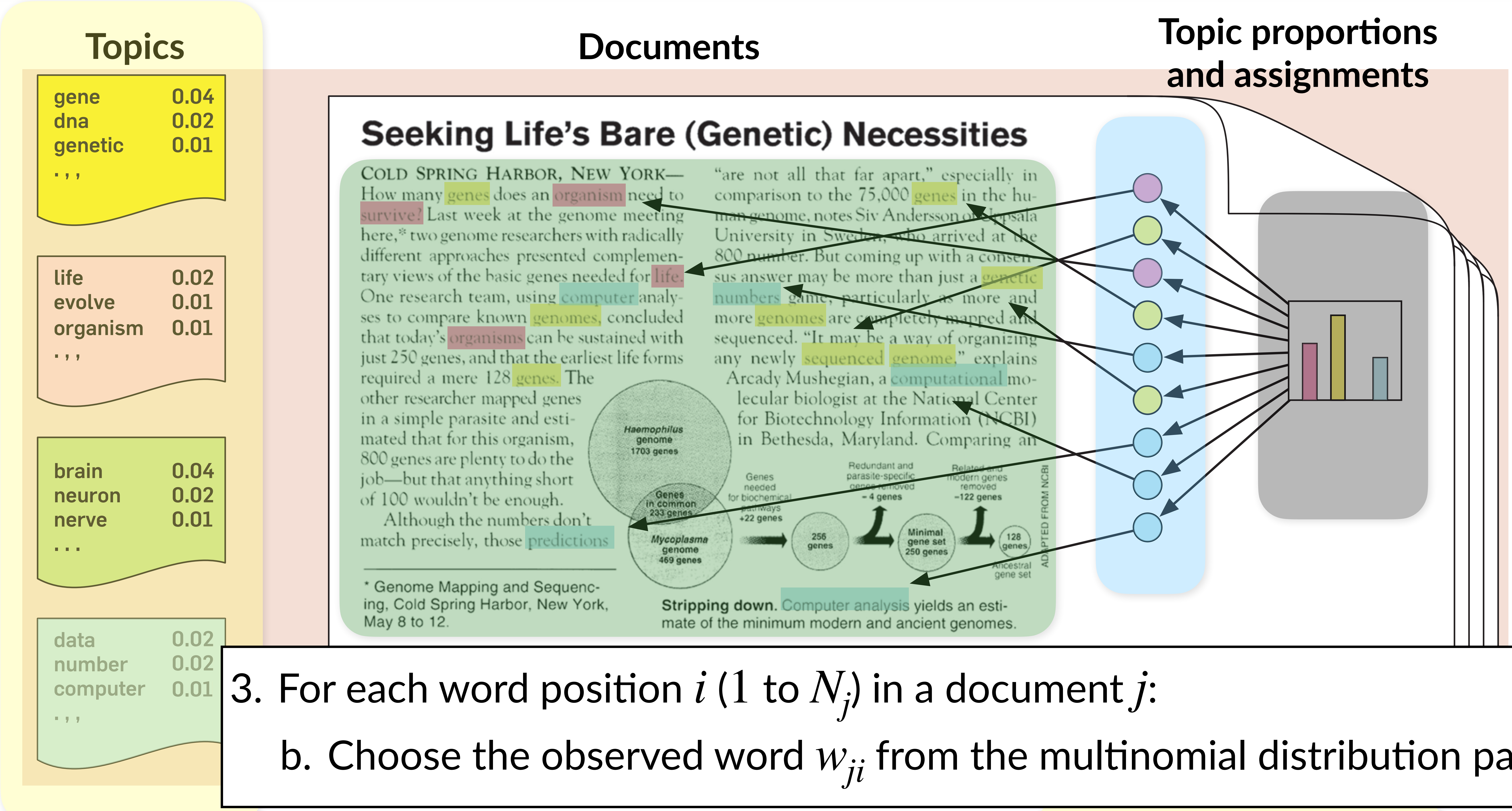
Stripping down. Computer analysis yields an estimate of the minimum modern and ancient genomes.

* Genome Mapping and Sequencing, Cold Spring Harbor, New York, May 8 to 12.

3. For each word position i (1 to N_j) in a document j:
   a. Select a latent topic z_{ji} from the multinomial distribution parametrised by $\boldsymbol{\theta}_j$



LDA — Generative story


Topics
gene 0.04
dna 0.02
genetic 0.01
. , ,
life 0.02
evolve 0.01
organism 0.01
. , ,
brain 0.04
neuron 0.02
nerve 0.01
. . .
data 0.02
number 0.02
computer 0.01
. , ,
Documents
Topic proportions and assignments
Haemophilus genome 1703 genes
Genes in common 233 genes
Mycoplasma genome 469 genes
Genes needed for biochemical pathways +22 genes
256 genes
Redundant and parasite-specific genes removed −4 genes
Minimal gene set 250 genes
Related and modern genes removed −122 genes
128 genes
Ancestral gene set
ADAPTED FROM NCBI

Seeking Life's Bare (Genetic) Necessities

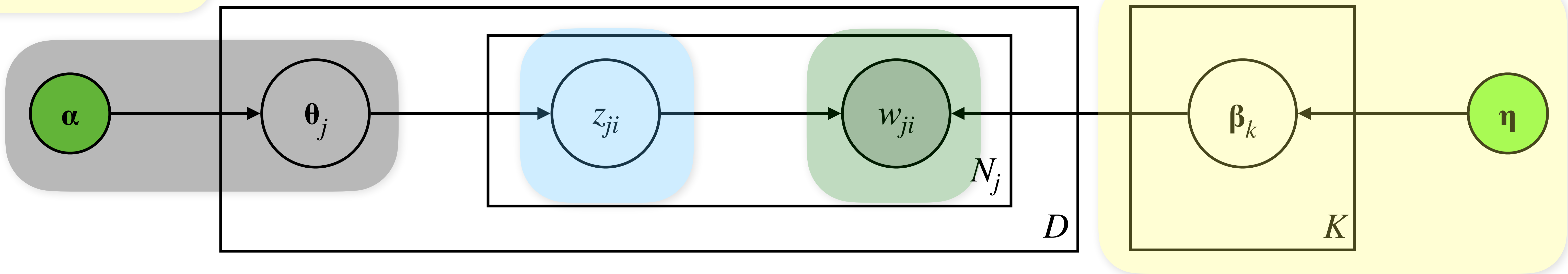
COLD SPRING HARBOR, NEW YORK— How many genes does an organism need to survive? Last week at the genome meeting here,* two genome researchers with radically different approaches presented complementary views of the basic genes needed for life. One research team, using computer analyses to compare known genomes, concluded that today's organisms can be sustained with just 250 genes, and that the earliest life forms required a mere 128 genes. The other researcher mapped genes in a simple parasite and estimated that for this organism, 800 genes are plenty to do the job—but that anything short of 100 wouldn't be enough.

Although the numbers don't match precisely, those predictions "are not all that far apart," especially in comparison to the 75,000 genes in the human genome, notes Siv Andersson of Uppsala University in Sweden, who arrived at the 800 number. But coming up with a consensus answer may be more than just a genetic numbers game, particularly as more and more genomes are completely mapped and sequenced. "It may be a way of organizing any newly sequenced genome," explains Arcady Mushegian, a computational molecular biologist at the National Center for Biotechnology Information (NCBI) in Bethesda, Maryland. Comparing an

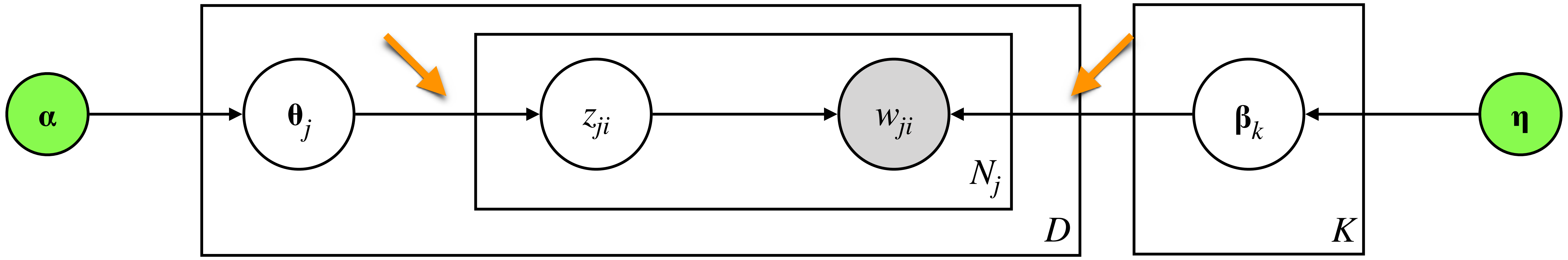
Stripping down. Computer analysis yields an estimate of the minimum modern and ancient genomes.

* Genome Mapping and Sequencing, Cold Spring Harbor, New York, May 8 to 12.

3. For each word position i (1 to N_j) in a document j:
    b. Choose the observed word w_{ji} from the multinomial distribution parametrised by $\boldsymbol{\beta}_{z_{ji}}$



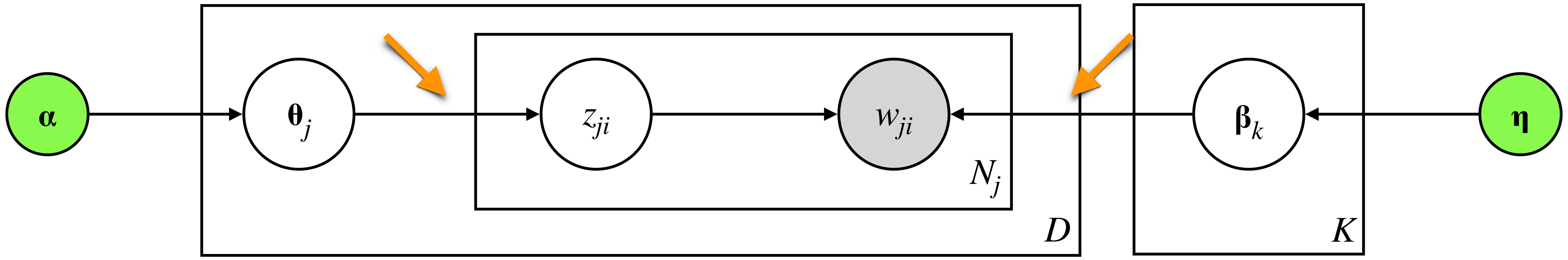
LDA — Multinomial distribution (Mult)



What is the probability of a set of outcomes for an event that has multiple outcomes?

– Roll a 6-sided dice 5 times. What is the probability of getting a “3” 1 time and a “6” 4 times?

LDA — Multinomial distribution (Mult)

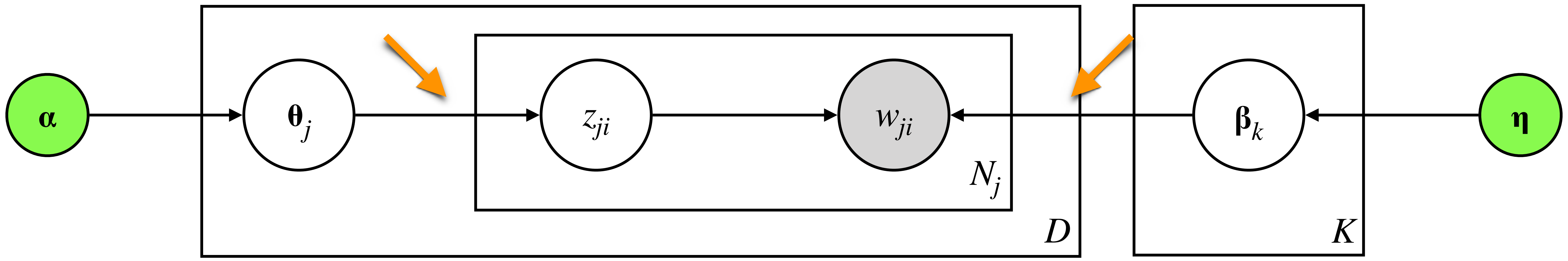


What is the probability of a set of outcomes for an event that has multiple outcomes?

– Roll a 6-sided dice 5 times. What is the probability of getting a "3" 1 time and a "6" 4 times?

$$\underbrace{\frac{5!}{1!4!}}_{\text{\#ways to get 1 “3” and 4 “6”s}} \cdot \underbrace{\left(\frac{1}{6}\right)}_{\text{prob. of 1 “3”}} \cdot \underbrace{\left(\frac{1}{6}\right)^4}_{\text{prob. of 4 “6”s}} \approx 0.00064$$

LDA — Multinomial distribution (Mult)



What is the probability of a set of outcomes for an event that has multiple outcomes?

– Roll a 6-sided dice 5 times. What is the probability of getting a “3” 1 time and a “6” 4 times?

$$\frac{5!}{1!4!} \cdot \left(\frac{1}{6}\right) \cdot \left(\frac{1}{6}\right)^4 \approx 0.00064$$

#ways to get 1 “3” and 4 “6”s

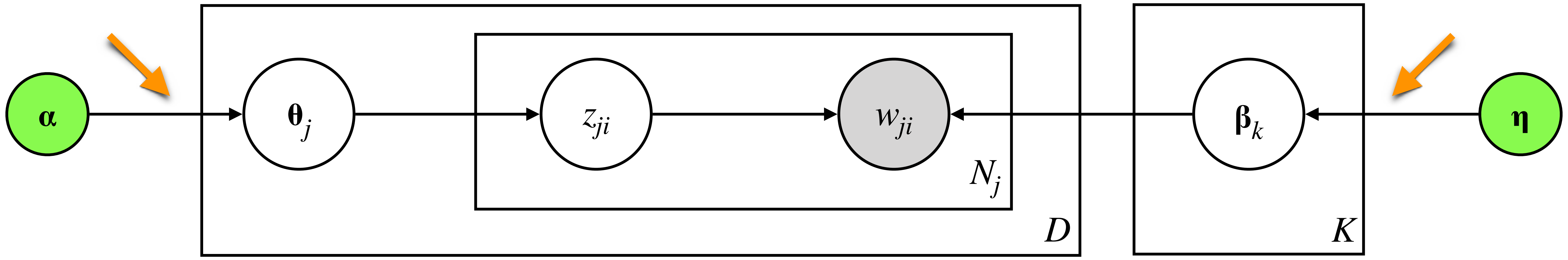
prob. of 1 “3”

prob. of 4 “6”s

youtube.com/watch?v=5A_H1eHbOCY
10 min explanation by Prof. John Tsitsiklis

Formally: $$p(n_1, \ldots, n_k) = \frac{n!}{n_1! \cdot \ldots \cdot n_k!} \cdot p_1^{n_1} \cdot \ldots \cdot p_k^{n_k} \text{ given } n, \{p_1, \ldots, p_k\}$$

LDA — Dirichlet distribution (Dir)



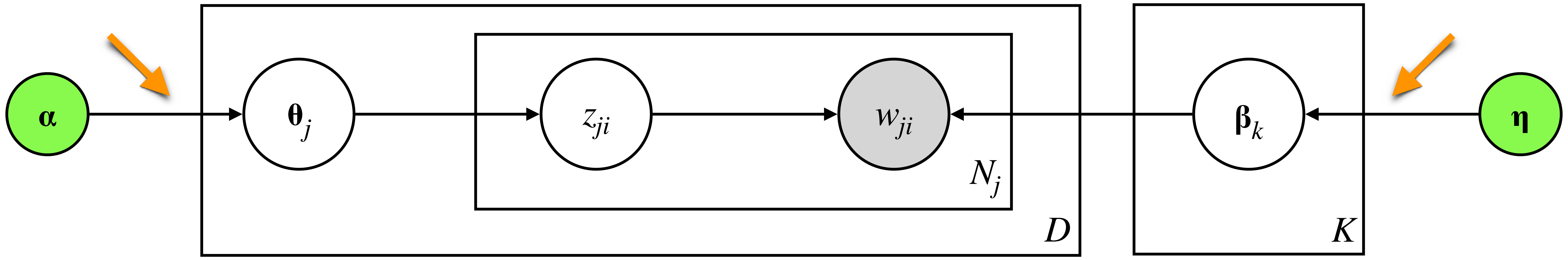
Dirichlet: Exponential family distribution over the simplex (positive vectors with elements that sum up to 1), essentially a distribution over multinomial distributions

$$p(\boldsymbol{\theta}|\boldsymbol{\alpha}) = \frac{\Gamma\left(\sum_{k=1}^{K}\alpha_k\right)}{\prod_{k=1}^{K}\Gamma\left(\alpha_k\right)} \cdot \prod_{k=1}^{K}\theta_k^{\alpha_k-1} \quad \text{where} \quad \Gamma(n) = (n-1)!$$

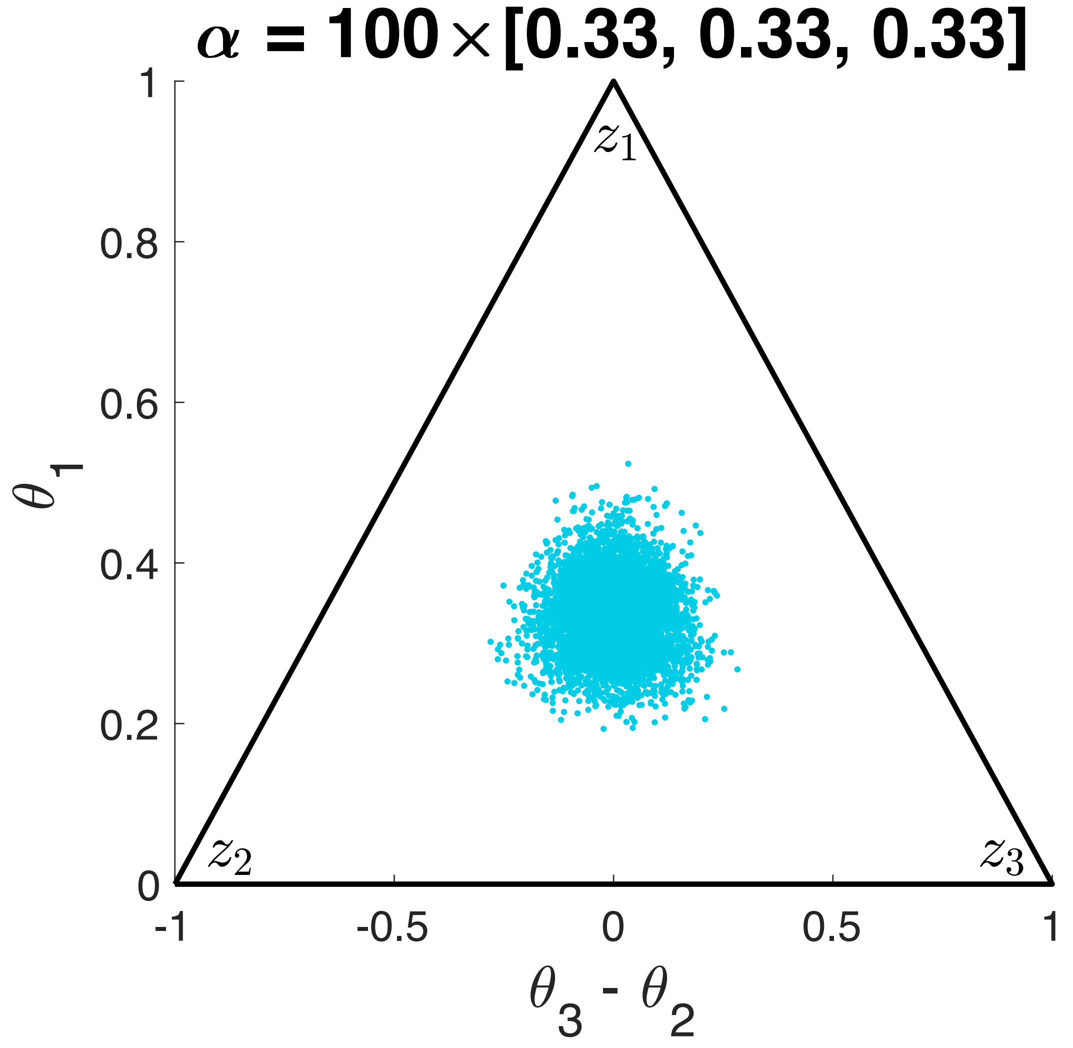
Parameter $\boldsymbol{\alpha}$ controls the mean shape and sparsity of $\boldsymbol{\theta}$ (same applies on $\boldsymbol{\eta}$ for $\boldsymbol{\beta}$)

Note: $\boldsymbol{\alpha}$ is a vector of K (= number of topics) parameters for $\boldsymbol{\theta}$ and $\boldsymbol{\eta}$ has V parameters for $\boldsymbol{\beta}$, where V is the size of the vocabulary (unique terms across all D documents)

LDA — Dirichlet distribution (Dir)

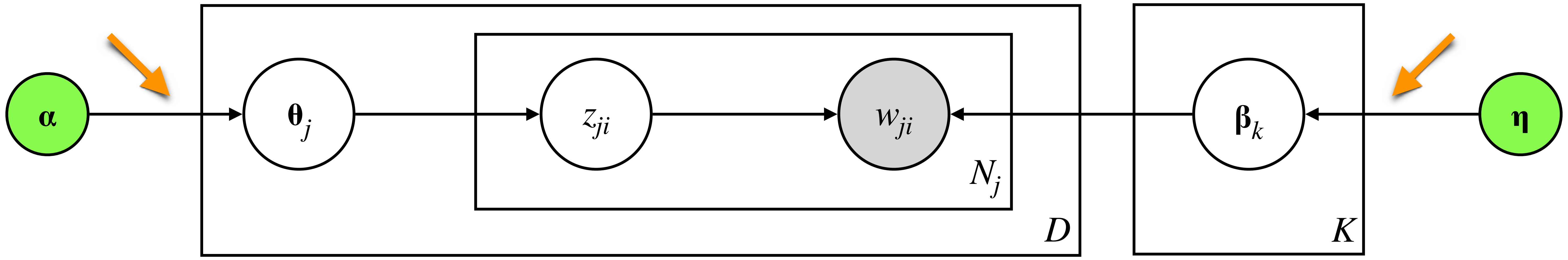


Assume a simplex $\boldsymbol{\theta} = \left[\theta_1, \theta_2, \theta_3\right]$ across $K = 3$ topics with $0 \leq \theta_i \leq 1$. How do different values for $\boldsymbol{\alpha}$ affect the $\boldsymbol{\theta}$ produced by the Dirichlet distribution? Let's plot $5{,}000$ samples for different $\boldsymbol{\alpha}$'s.

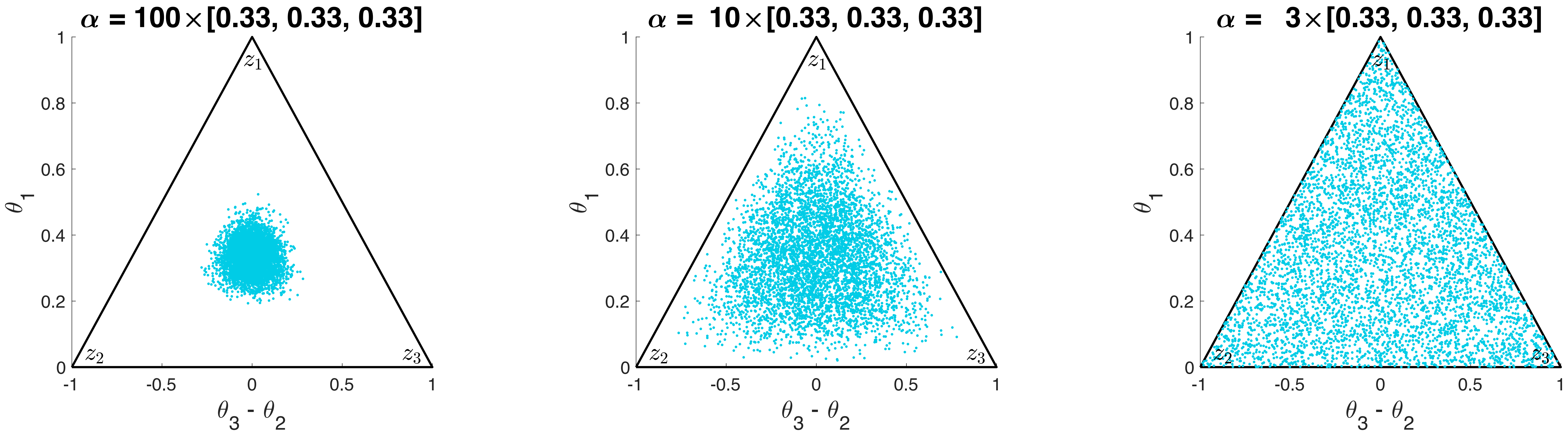


Large values of $\boldsymbol{\alpha}$ lead to more dense $\boldsymbol{\theta}$'s

LDA — Dirichlet distribution (Dir)

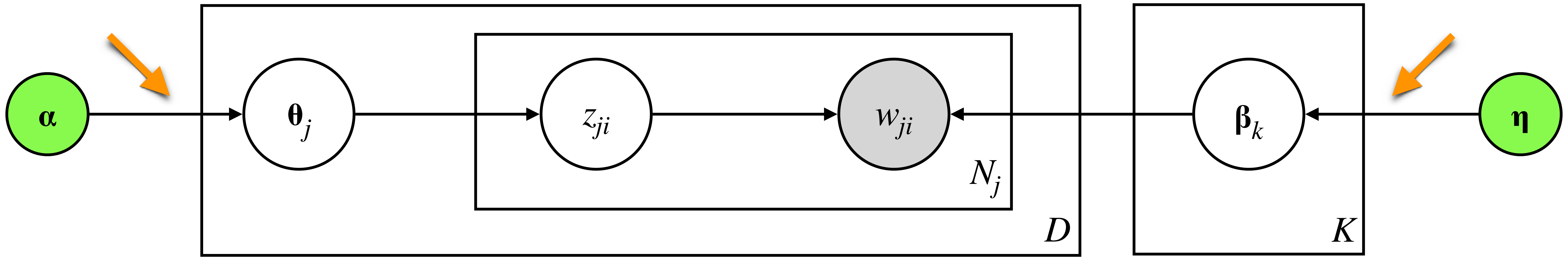


Assume a simplex $\boldsymbol{\theta} = [\theta_1, \theta_2, \theta_3]$ across $K = 3$ topics with $0 \leq \theta_i \leq 1$. How do different values for $\boldsymbol{\alpha}$ affect the $\boldsymbol{\theta}$ produced by the Dirichlet distribution? Let's plot $5{,}000$ samples for different $\boldsymbol{\alpha}$'s.

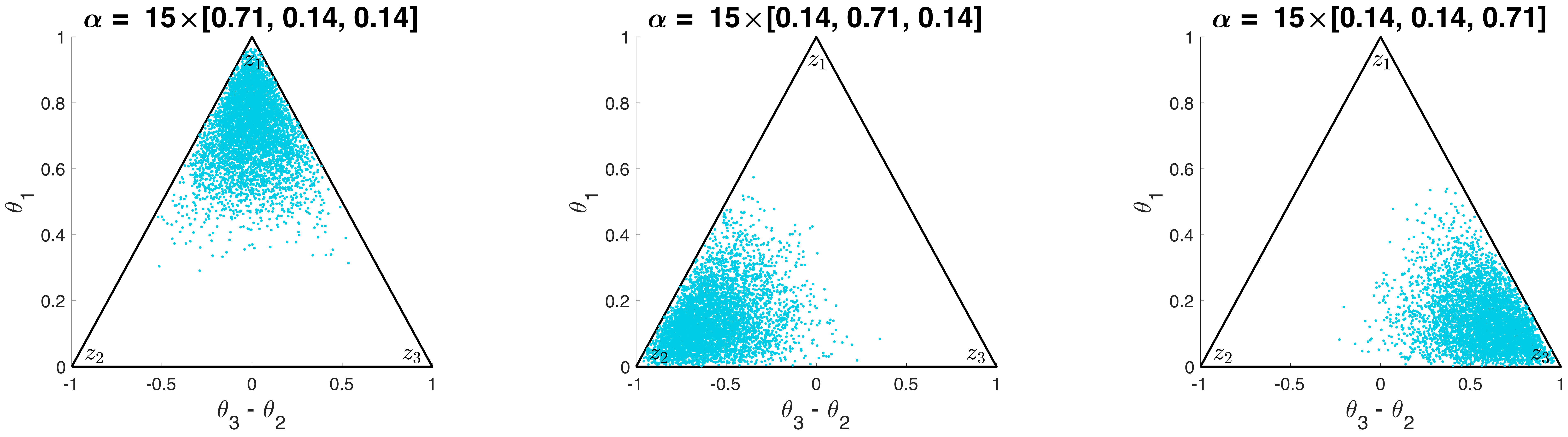


Large values of $\boldsymbol{\alpha}$ lead to more dense $\boldsymbol{\theta}$'s

LDA — Dirichlet distribution (Dir)

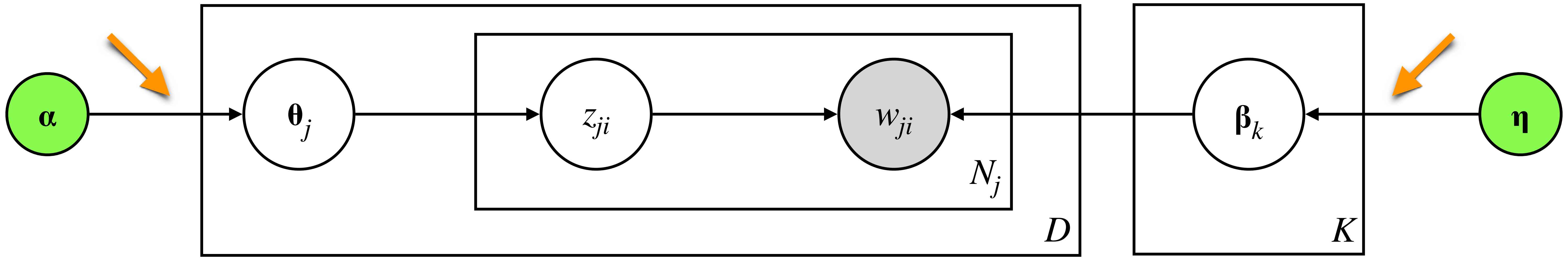


Assume a simplex $\boldsymbol{\theta} = [\theta_1, \theta_2, \theta_3]$ across $K = 3$ topics with $0 \leq \theta_i \leq 1$. How do different values for $\boldsymbol{\alpha}$ affect the $\boldsymbol{\theta}$ produced by the Dirichlet distribution? Let's plot $5{,}000$ samples for different $\boldsymbol{\alpha}$'s.

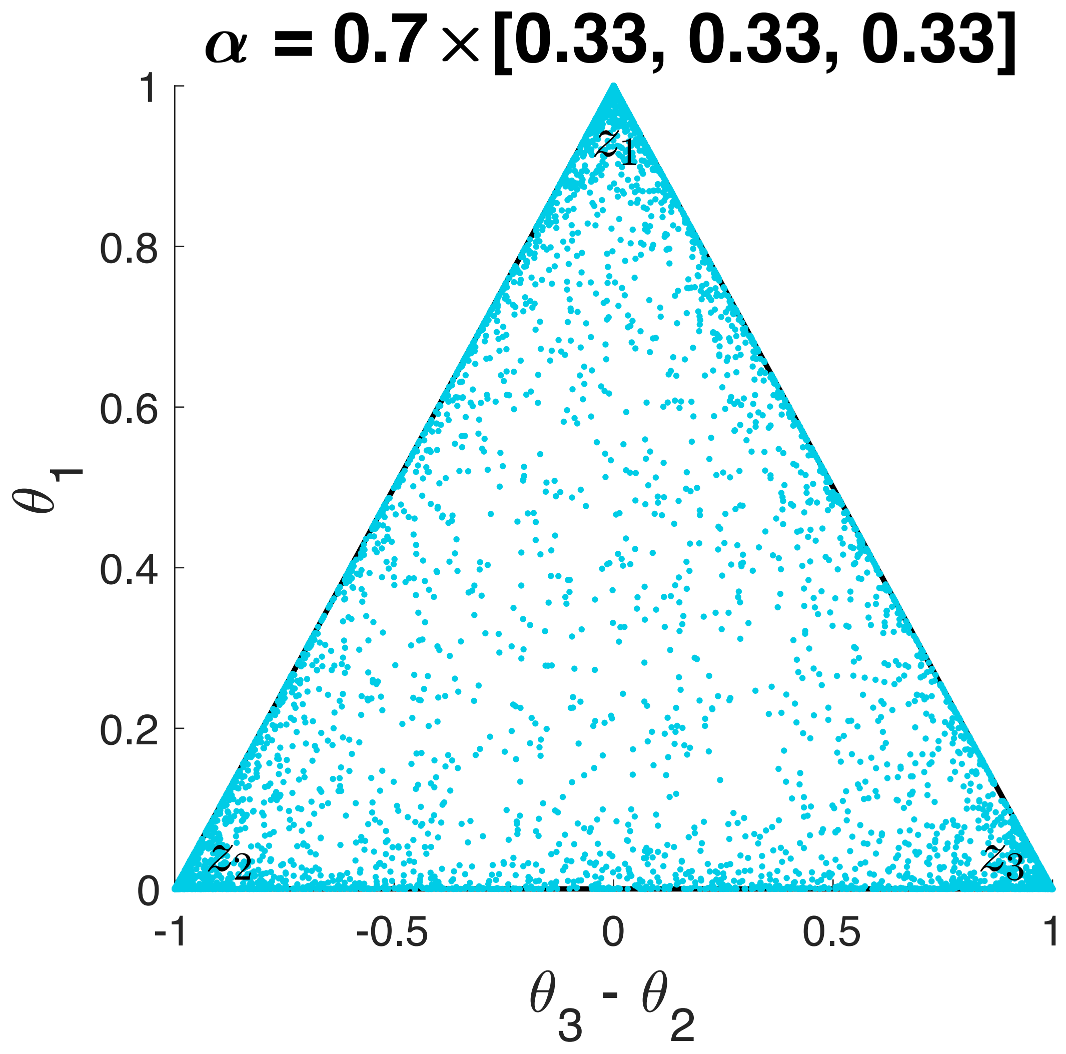


Imbalance in $\boldsymbol{\alpha}$ shapes the focus of the distribution

LDA — Dirichlet distribution (Dir)

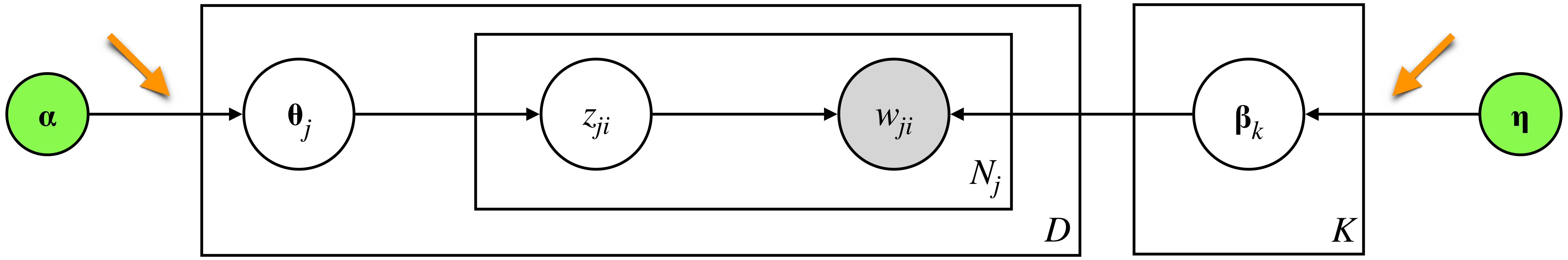


Assume a simplex $\boldsymbol{\theta} = \left[\theta_1, \theta_2, \theta_3\right]$ across $K = 3$ topics with $0 \leq \theta_i \leq 1$. How do different values for $\boldsymbol{\alpha}$ affect the $\boldsymbol{\theta}$ produced by the Dirichlet distribution? Let's plot $5{,}000$ samples for different $\boldsymbol{\alpha}$'s.

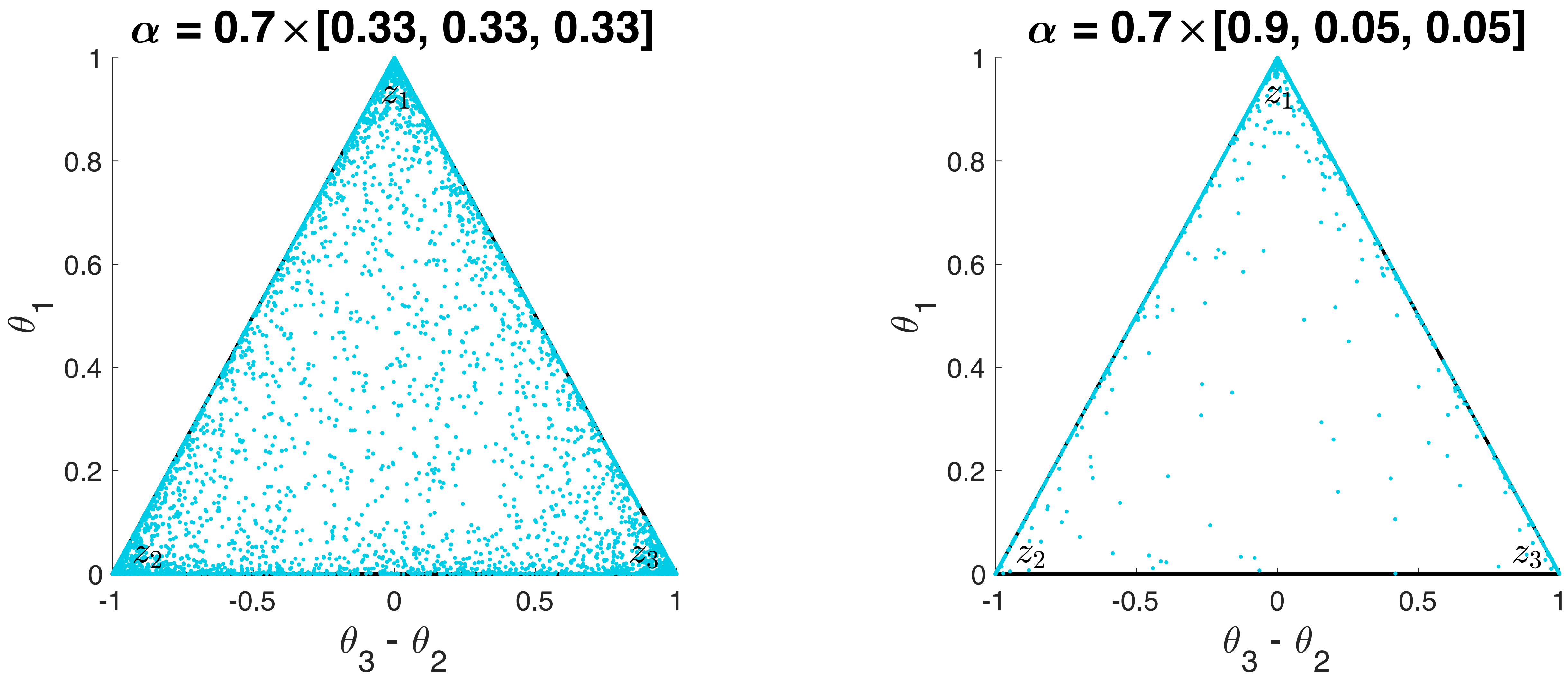


Values of $\boldsymbol{\alpha} < 1$ create increasingly sparse outputs

LDA — Dirichlet distribution (Dir)

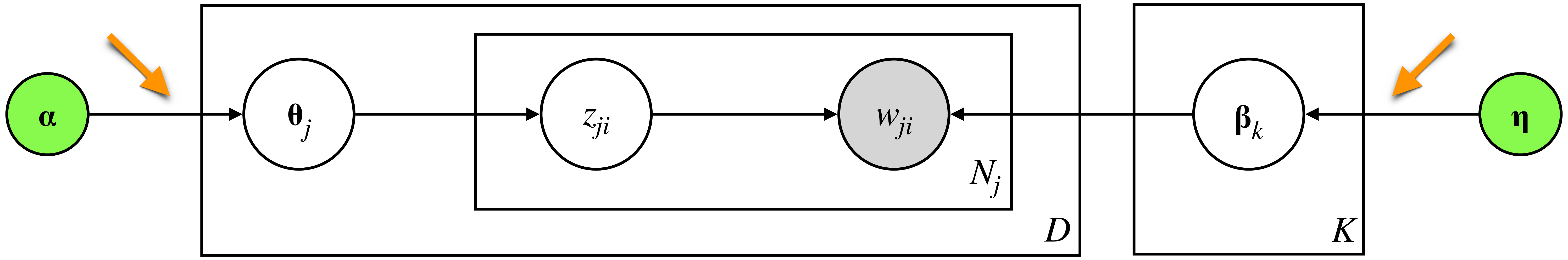


Assume a simplex $\boldsymbol{\theta} = [\theta_1, \theta_2, \theta_3]$ across $K = 3$ topics with $0 \le \theta_i \le 1$. How do different values for $\boldsymbol{\alpha}$ affect the $\boldsymbol{\theta}$ produced by the Dirichlet distribution? Let's plot $5{,}000$ samples for different $\boldsymbol{\alpha}$'s.

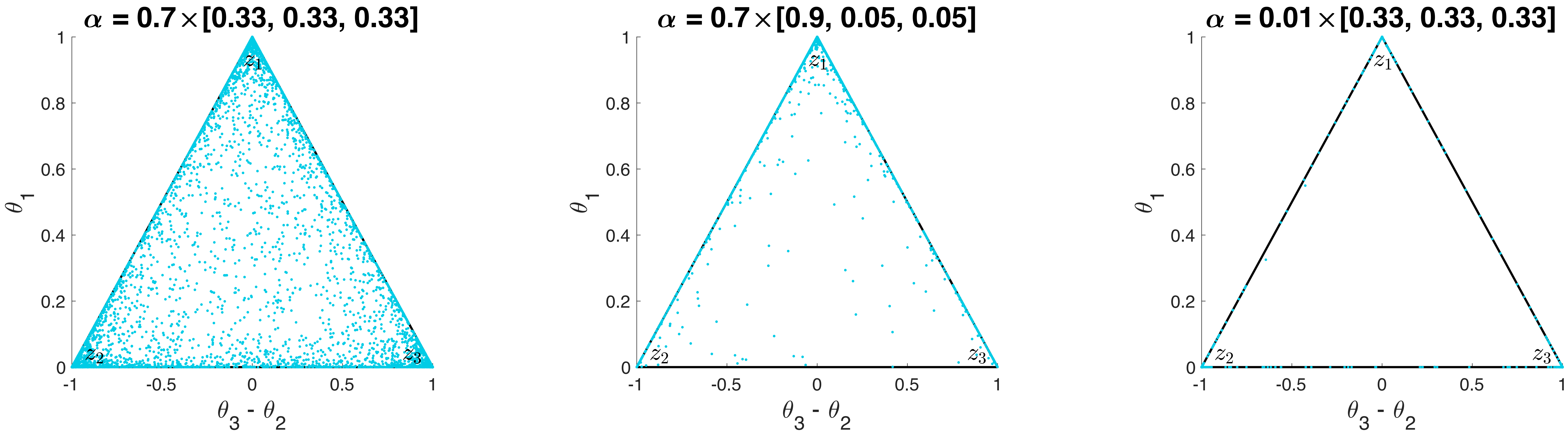


Values of $\boldsymbol{\alpha} < 1$ create increasingly sparse outputs

LDA — Dirichlet distribution (Dir)

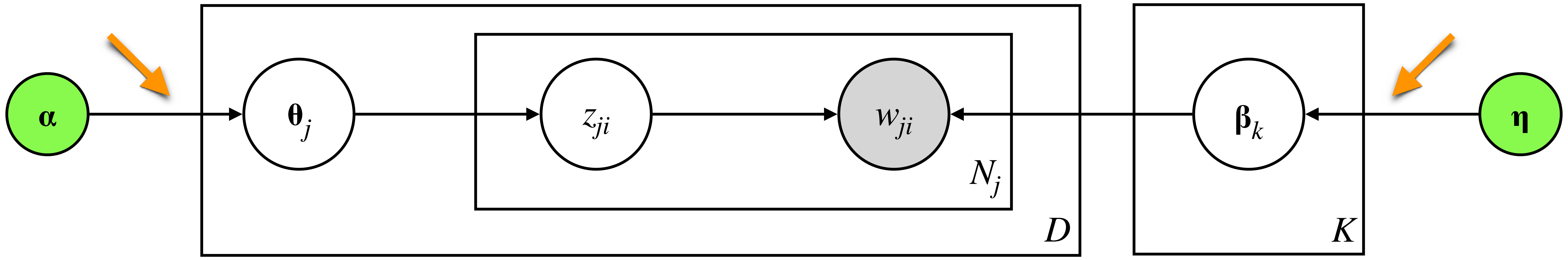


Assume a simplex $\boldsymbol{\theta} = [\theta_1, \theta_2, \theta_3]$ across $K = 3$ topics with $0 \leq \theta_i \leq 1$. How do different values for $\boldsymbol{\alpha}$ affect the $\boldsymbol{\theta}$ produced by the Dirichlet distribution? Let's plot $5{,}000$ samples for different $\boldsymbol{\alpha}$'s.



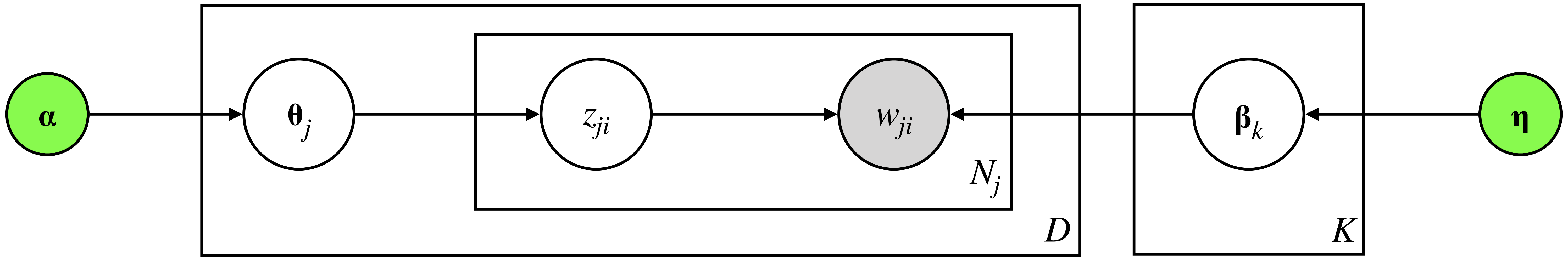
Values of $\boldsymbol{\alpha} < 1$ create increasingly sparse outputs

LDA — Why do we combine Mult and Dir distributions?



- The Dirichlet distribution is ***conjugate*** to the Multinomial distribution
- Posterior $p(\boldsymbol{\beta}|\boldsymbol{\eta}, \mathbf{w})$ and prior $p(\boldsymbol{\beta}|\boldsymbol{\eta})$ belong to the same distribution family as the prior (Dirichlet) given that $p(\mathbf{w}|\boldsymbol{\beta})$ is a Multinomial and $p(\boldsymbol{\beta}|\boldsymbol{\eta})$ a Dirichlet
- Abstracting the math, observed data ($\mathbf{w}$) are adding to our prior intuition ($\boldsymbol{\eta}$) about how words relate with topics

LDA — Inference



Joint probability distribution

$$p(\mathbf{W},\boldsymbol{\Theta},\mathbf{B},\mathbf{Z} \mid \boldsymbol{\alpha},\boldsymbol{\eta}) = \prod_{k=1}^{K} p(\boldsymbol{\beta}_k \mid \boldsymbol{\eta}) \prod_{j=1}^{D} p(\boldsymbol{\theta}_j \mid \boldsymbol{\alpha}) \left(\prod_{i=1}^{N_j} p(z_{ji} \mid \boldsymbol{\theta}_j)\, p(w_{ji} \mid \mathbf{B}, z_{ji}) \right)$$

We are interested in this posterior $$p(\boldsymbol{\Theta},\mathbf{B},\mathbf{Z} \mid \mathbf{W},\boldsymbol{\alpha},\boldsymbol{\eta}) = \frac{p(\boldsymbol{\Theta},\mathbf{B},\mathbf{Z},\mathbf{W} \mid \boldsymbol{\alpha},\boldsymbol{\eta})}{\int_{\mathbf{B}} \int_{\boldsymbol{\Theta}} \sum_{z} p(\boldsymbol{\Theta},\mathbf{B},\mathbf{Z},\mathbf{W} \mid \boldsymbol{\alpha},\boldsymbol{\eta})}$$

can’t compute → approximate inference

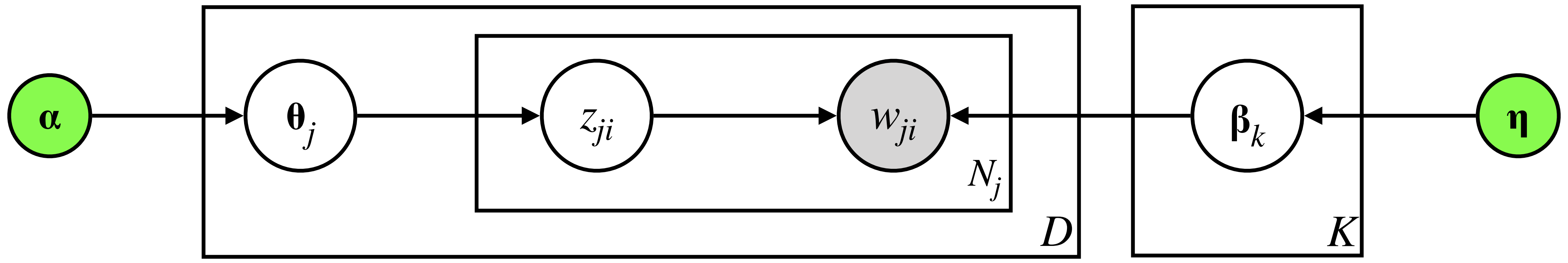
LDA — Inference, Gibbs sampling

- Initialise probabilities randomly or uniformly (*assume that we know everything!*)
- In each step, replace the value of one of the variables by a value drawn from the distribution of that variable conditioned on the values of the remaining variables
- Repeat until convergence

```
For $t = 1, \dots, T$:
    Sample $x_1^{(t+1)} \sim p\left(x_1 \,|\, x_2^{(t)}, \dots, x_N^{(t)}\right)$
    Sample $x_2^{(t+1)} \sim p\left(x_2 \,|\, x_1^{(t+1)}, x_3^{(t)}, \dots, x_N^{(t)}\right)$
    $\dots$
    Sample $x_j^{(t+1)} \sim p\left(x_j \,|\, x_1^{(t+1)}, x_2^{(t+1)}, \dots, x_{j-1}^{(t+1)}, x_{j+1}^{(t)}, \dots, x_N^{(t)}\right)$
    $\dots$
    Sample $x_N^{(t+1)} \sim p\left(x_N \,|\, x_1^{(t+1)}, \dots, x_{N-1}^{(t+1)}\right)$
```

LDA — Inference, Gibbs sampling

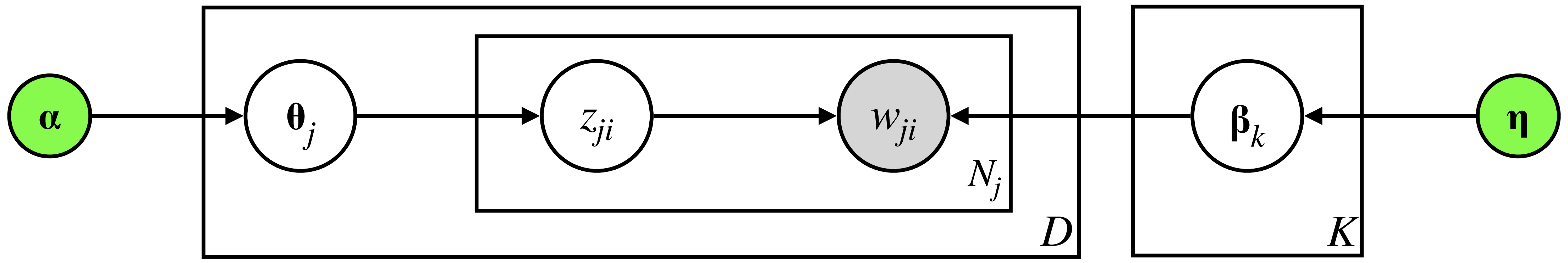
- Initialise probabilities randomly or uniformly
- Go over each word i in every document j (w_{ji})
- Estimate the probability of assigning w_{ji} to each topic, conditioned on the topic assignments ($\mathbf{z}_{j,-i}$) of all other words $\mathbf{w}_{j,-i}$ (*notation indicating the exclusion of* w_{ji})



$$p(z_{ji} = k \,|\, \mathbf{z}_{j,-i}, \mathbf{W}, \boldsymbol{\alpha}, \boldsymbol{\eta}) \;\propto \frac{n_{j,k,-i} + \alpha_k}{\sum_{k'=1}^{K} n_{j,k',-i} + \alpha_{k'}} \cdot \frac{m_{k,w_{ji},-i} + \eta_{w_{ji}}}{\sum_{\nu=1}^{V} m_{k,\nu,-i} + \eta_\nu}$$

LDA — Inference, Gibbs sampling

- Initialise probabilities randomly or uniformly
- Go over each word i in every document j (w_{ji})
- Estimate the probability of assigning w_{ji} to each topic, conditioned on the topic assignments ($\mathbf{z}_{j,-i}$) of all other words $\mathbf{w}_{j,-i}$ (*notation indicating the exclusion of* w_{ji})



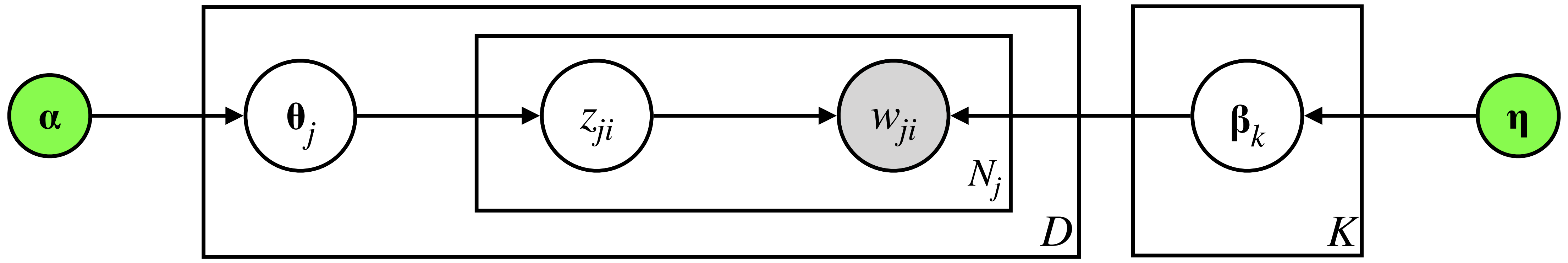
$$p(z_{ji} = k \,|\, \mathbf{z}_{j,-i}, \mathbf{W}, \boldsymbol{\alpha}, \boldsymbol{\eta}) \propto \boxed{\frac{n_{j,k,-i} + \alpha_k}{\sum_{k'=1}^{K} n_{j,k',-i} + \alpha_{k'}}} \cdot \boxed{\frac{m_{k,w_{ji},-i} + \eta_{w_{ji}}}{\sum_{v=1}^{V} m_{k,v,-i} + \eta_v}}$$

How much does document j "like" topic k?

How much does topic k "like" word w_{ji}?

LDA — Inference, Gibbs sampling

- Initialise probabilities randomly or uniformly
- Go over each word i in every document j (w_{ji})
- Estimate the probability of assigning w_{ji} to each topic, conditioned on the topic assignments ($\mathbf{z}_{j,-i}$) of all other words $\mathbf{w}_{j,-i}$ (*notation indicating the exclusion of* w_{ji})



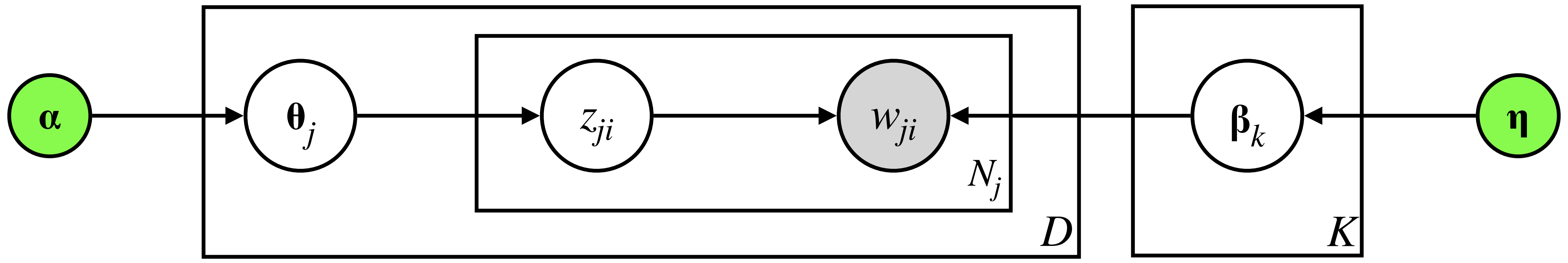
$$p(z_{ji} = k \,|\, \mathbf{z}_{j,-i}, \mathbf{W}, \boldsymbol{\alpha}, \boldsymbol{\eta}) \propto \frac{\boxed{n_{j,k,-i} + \alpha_k}}{\sum_{k'=1}^{K} n_{j,k',-i} + \alpha_{k'}} \cdot \frac{\boxed{m_{k,w_{ji},-i} + \eta_{w_{ji}}}}{\sum_{v=1}^{V} m_{k,v,-i} + \eta_v}$$

# topic k is assigned to a word in document j without counting the current word

# word w_{ji} is associated with topic k in all documents without counting the current instance of w_{ji}

LDA — Inference, Gibbs sampling

- Initialise probabilities randomly or uniformly
- Go over each word i in every document j (w_{ji})
- Estimate the probability of assigning w_{ji} to each topic, conditioned on the topic assignments ($\mathbf{z}_{j,-i}$) of all other words $\mathbf{w}_{j,-i}$ (*notation indicating the exclusion of* w_{ji})



$$p(z_{ji} = k \,|\, \mathbf{z}_{j,-i}, \mathbf{W}, \boldsymbol{\alpha}, \boldsymbol{\eta}) \propto \boxed{\frac{n_{j,k,-i} + \alpha_k}{\sum_{k'=1}^{K} n_{j,k',-i} + \alpha_{k'}}} \cdot \boxed{\frac{m_{k,w_{ji},-i} + \eta_{w_{ji}}}{\sum_{v=1}^{V} m_{k,v,-i} + \eta_v}}$$

How much does topic k “like” word w_{ji}?

How much does document j “like” topic k?

- From the above conditional distribution, sample a topic and set it as the new topic assignment z_{ji} of w_{ji}

LDA — Gibbs sampling, toy example

- **Consider $K = 3$ topics**
- ...

LDA — Gibbs sampling, toy example

- Consider $K = 3$ topics
- **Sampling from document** j (*word order doesn't matter*)
- …

document j

$\mathbf{z_{ji}}$	?	?	?	?	?
$\mathbf{w_{ji}}$	Brexit	deficit	Europe	market	single

LDA — Gibbs sampling, toy example

- Consider $K = 3$ topics
- Sampling from document j (*word order doesn't matter*)
- **Randomly assign topics to all words in document** j (*and all other docs*)
- ...

document j

$\mathbf{z_{ji}}$	$\mathbf{3}$	**?**	**?**	**?**	**?**
$\mathbf{w_{ji}}$	Brexit	deficit	Europe	market	single

LDA — Gibbs sampling, toy example

- Consider $K = 3$ topics
- Sampling from document j (*word order doesn't matter*)
- **Randomly assign topics to all words in document** j (*and all other docs*)
- ...

document j

$\mathbf{z_{ji}}$	**3**	**2**	**?**	**?**	**?**
$\mathbf{w_{ji}}$	Brexit	deficit	Europe	market	single

LDA — Gibbs sampling, toy example

- Consider $K = 3$ topics
- Sampling from document j (*word order doesn't matter*)
- **Randomly assign topics to all words in document** j (*and all other docs*)
- …

document j

$\mathbf{z_{ji}}$	**3**	**2**	**3**	**1**	**1**
$\mathbf{w_{ji}}$	Brexit	deficit	Europe	market	single

LDA — Gibbs sampling, toy example

- Consider $K = 3$ topics
- Sampling from document j (*word order doesn't matter*)
- Randomly assign topics to all words in document j (*and all other docs*)
- **Update the word-topic counts for all documents**
- …

document j

z_{ji}	3	2	3	1	1
w_{ji}	Brexit	deficit	Europe	market	single

word-topic counts across all documents

words / topics	1	2	3
Brexit	100	30	2
deficit	10	60	0
Europe	95	5	2
market	50	70	5
single	50	15	90
…	…	…	…

LDA — Gibbs sampling, toy example

- Consider $K = 3$ topics
- Sampling from document j (*word order doesn't matter*)
- Randomly assign topics to all words in document j (*and all other docs*)
- Update the word-topic counts for all documents
- **Sample the first word ("Brexit") in document j; unassign it from topic 3 and decrement its count in the word-topic counts**
- …

document j

z_{ji}	3	2	3	1	1
w_{ji}	Brexit	deficit	Europe	market	single

word-topic counts across all documents

words / topics	1	2	3
Brexit	100	30	2
deficit	10	60	0
Europe	95	5	2
market	50	70	5
single	50	15	90
…	…	…	…

LDA — Gibbs sampling, toy example

- Consider $K = 3$ topics
- Sampling from document j (*word order doesn't matter*)
- Randomly assign topics to all words in document j (*and all other docs*)
- Update the word-topic counts for all documents
- **Sample the first word ("Brexit") in document j; unassign it from topic 3 and decrement its count in the word-topic counts**
- …

document j

$\mathbf{z_{ji}}$	?	2	3	1	1
$\mathbf{w_{ji}}$	Brexit	deficit	Europe	market	single

word-topic counts across all documents

words / topics	1	2	3
Brexit	100	30	2 – 1
deficit	10	60	0
Europe	95	5	2
market	50	70	5
single	50	15	90
…	…	…	…

LDA — Gibbs sampling, toy example

- Consider $K = 3$ topics
- Sampling from document j (*word order doesn't matter*)
- Randomly assign topics to all words in document j (*and all other docs*)
- Update the word-topic counts for all documents
- **Sample the first word ("Brexit") in document j; unassign it from topic 3 and decrement its count in the word-topic counts**
- ...

document j

$\mathbf{z_{ji}}$	?	2	3	1	1
$\mathbf{w_{ji}}$	Brexit	deficit	Europe	market	single

word-topic counts across all documents

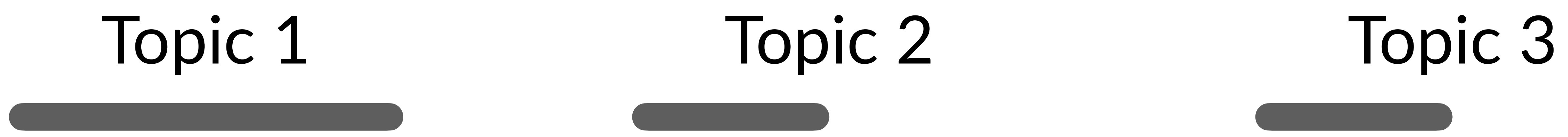
words / topics	1	2	3
Brexit	100	30	1
deficit	10	60	0
Europe	95	5	2
market	50	70	5
single	50	15	90
...	...	...	...

LDA — Gibbs sampling, toy example

- …
- Randomly assign topics to all words in document j (*and all other docs*)
- Update the word-topic counts for all documents
- Sample the first word (“Brexit”) in document j; unassign it from topic 3 and decrement its count in the word-topic counts
- **What are the revised topic proportions in document j?**
- …

document j

$\mathbf{z_{ji}}$	?	2	3	1	1
$\mathbf{w_{ji}}$	Brexit	deficit	Europe	market	single



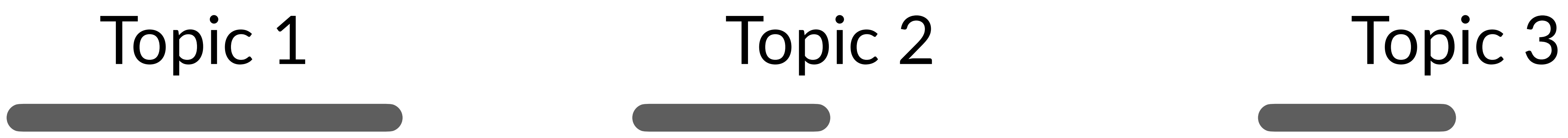
$$p(z_{ji} = k \,|\, \mathbf{z}_{j,-i}, \mathbf{w}, \boldsymbol{\alpha}, \boldsymbol{\eta}) \propto \boxed{\frac{n_{j,k,-i} + \alpha_k}{\sum_{k'=1}^{K} n_{j,k',-i} + \alpha_{k'}}} \cdot \frac{m_{k,w_{ji},-i} + \eta_{w_{ji}}}{\sum_{v=1}^{V} m_{k,v,-i} + \eta_v}$$

LDA — Gibbs sampling, toy example

- …
- Update the word-topic counts for all documents
- Sample the first word (“Brexit”) in document j; unassign it from topic 3 and decrement its count in the word-topic counts
- What are the revised topic proportions in document j?
- **How much does each topic “like” the word Brexit?**
- …

document j

z_{ji}	?	2	3	1	1
w_{ji}	Brexit	deficit	Europe	market	single



words / topics	1	2	3
Brexit	100	30	1
deficit	10	60	0
Europe	95	5	2
market	50	70	5
single	50	15	90
…	…	…	…

$$p(z_{ji}=k\,|\,\mathbf{z}_{j,-i},\mathbf{w},\boldsymbol{\alpha},\boldsymbol{\eta}) \propto \frac{n_{j,k,-i}+\alpha_k}{\sum_{k'=1}^{K} n_{j,k',-i}+\alpha_{k'}} \cdot \boxed{\frac{m_{k,w_{ji},-i}+\eta_{w_{ji}}}{\sum_{\nu=1}^{V} m_{k,\nu,-i}+\eta_\nu}}$$

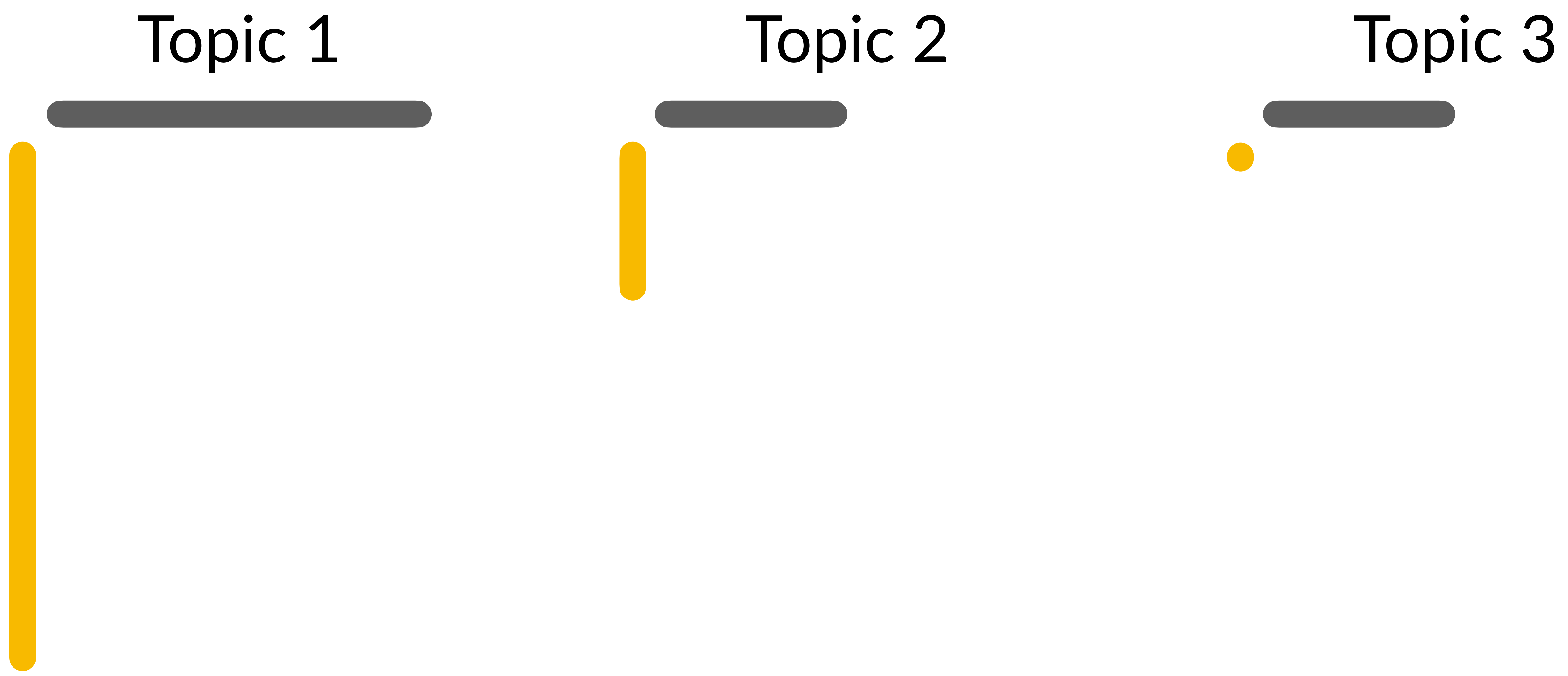
LDA — Gibbs sampling, toy example

- …
- Update the word-topic counts for all documents
- Sample the first word (“Brexit”) in document j; unassign it from topic 3 and decrement its count in the word-topic counts
- What are the revised topic proportions in document j?
- **How much does each topic “like” the word Brexit?**
- …

document j

z_{ji}	?	2	3	1	1
w_{ji}	Brexit	deficit	Europe	market	single

words / topics	1	2	3
Brexit	100	30	1
deficit	10	60	0
Europe	95	5	2
market	50	70	5
single	50	15	90
…	…	…	…



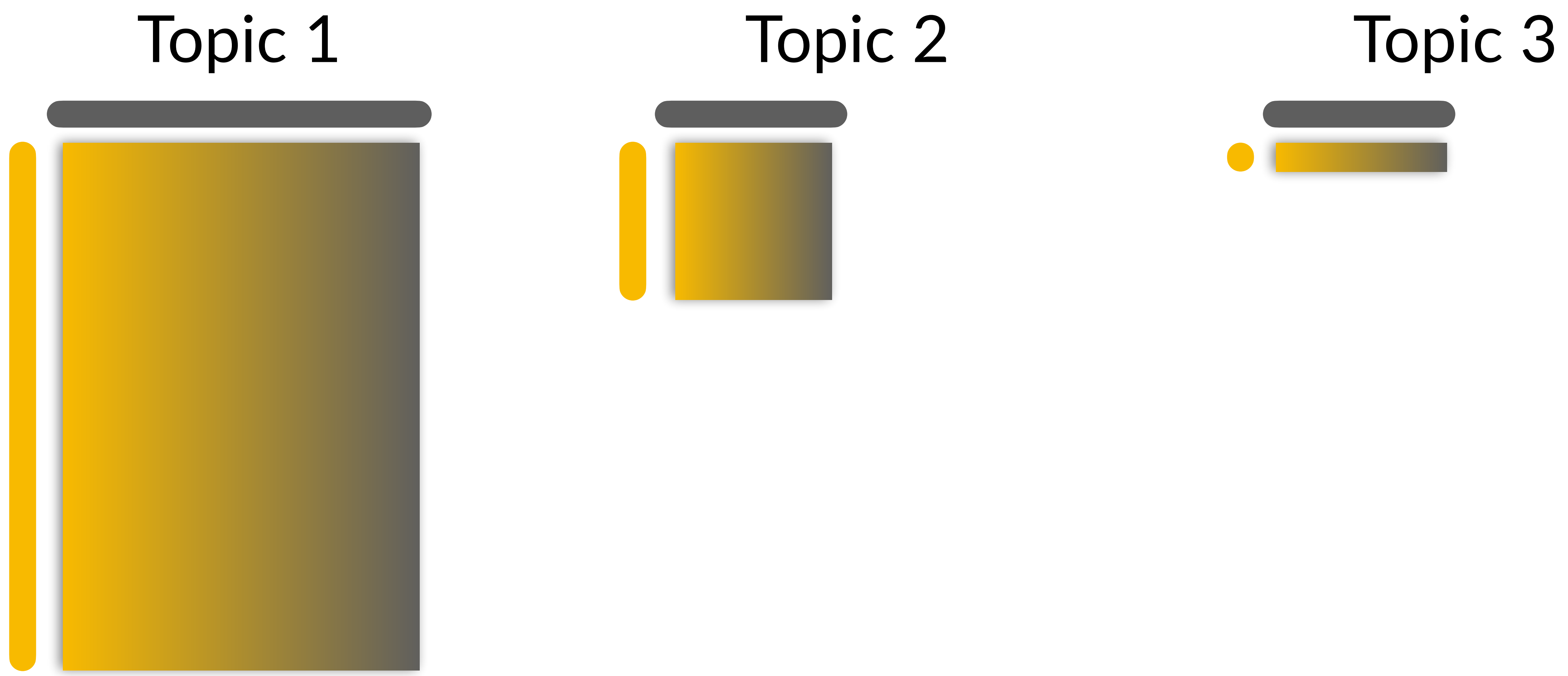
LDA — Gibbs sampling, toy example

- …
- Update the word-topic counts for all documents
- Sample the first word (“Brexit”) in document j; unassign it from topic 3 and decrement its count in the word-topic counts
- What are the revised topic proportions in document j?
- **How much does each topic “like” the word Brexit?**
- …

document j

z_{ji}	?	2	3	1	1
w_{ji}	Brexit	deficit	Europe	market	single

words / topics	1	2	3
Brexit	100	30	1
deficit	10	60	0
Europe	95	5	2
market	50	70	5
single	50	15	90
…	…	…	…



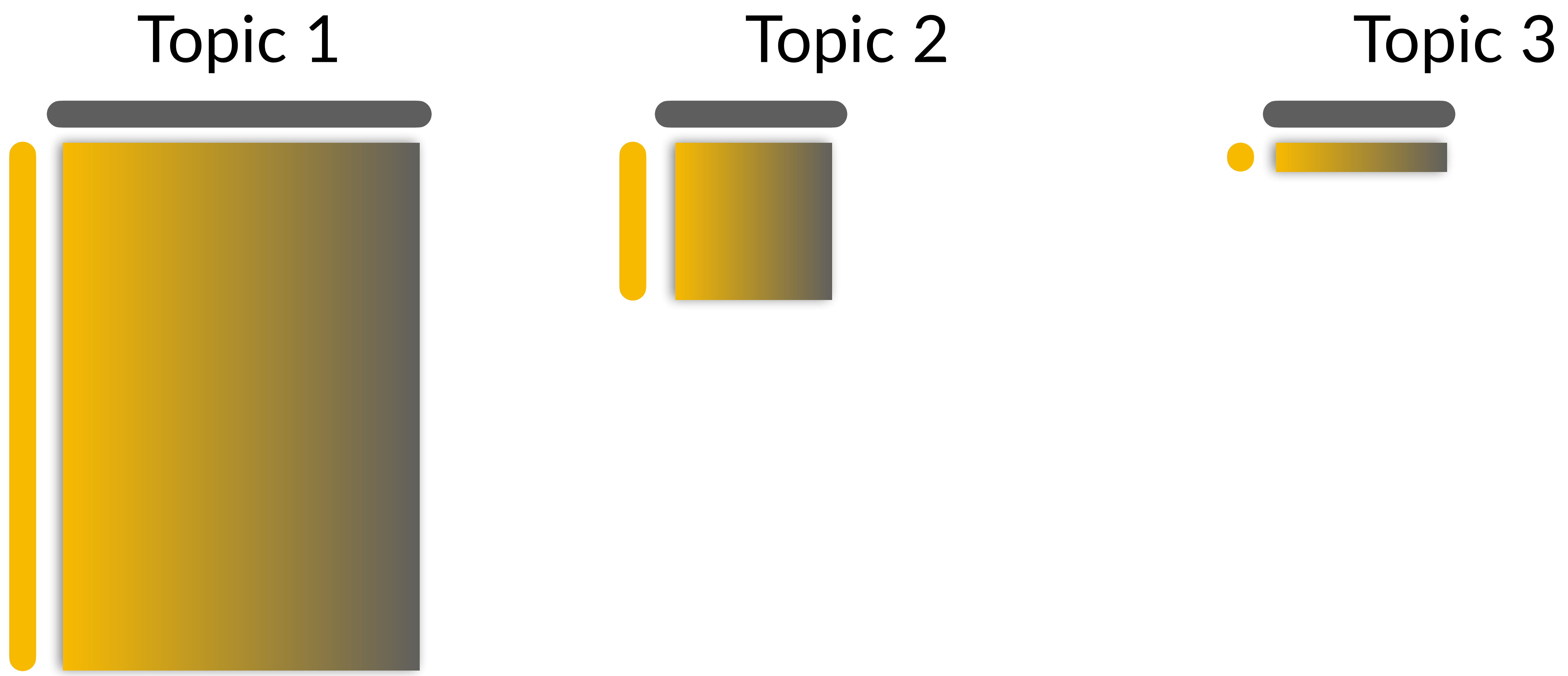
LDA — Gibbs sampling, toy example

- …
- Sample the first word (“Brexit”) in document j; unassign it from topic 3 and decrement its count in the word-topic counts
- What are the revised topic proportions in document j?
- How much does each topic “like” the word Brexit?
- **Sample from the revised conditional distribution** $p(z_{ji} = k \,|\, \mathbf{z}_{j,-i}, \mathbf{W}, \boldsymbol{\alpha}, \boldsymbol{\eta})$
- …

document j

z_{ji}	?	2	3	1	1
w_{ji}	Brexit	deficit	Europe	market	single

words / topics	1	2	3
Brexit	100	30	1
deficit	10	60	0
Europe	95	5	2
market	50	70	5
single	50	15	90
…	…	…	…



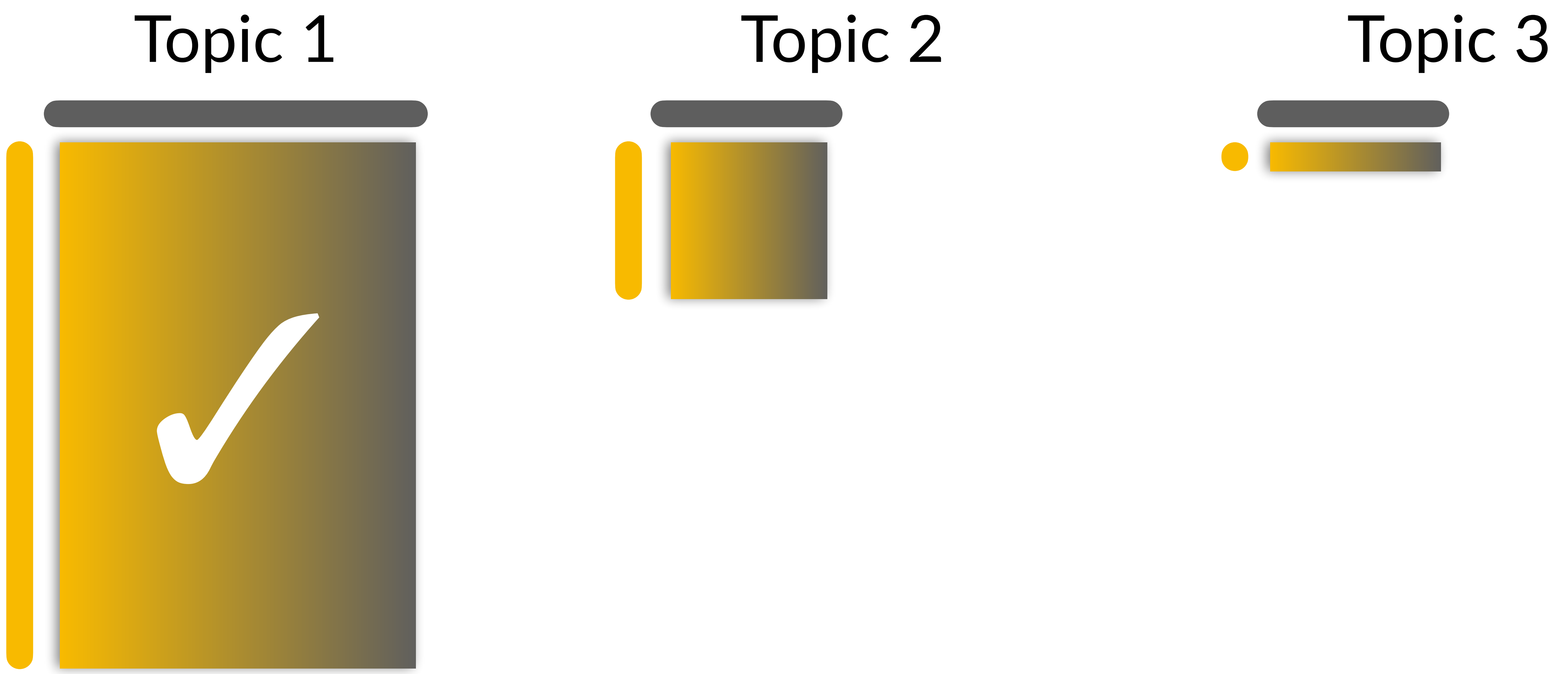
LDA — Gibbs sampling, toy example

- …
- Sample the first word (“Brexit”) in document j; unassign it from topic 3 and decrement its count in the word-topic counts
- What are the revised topic proportions in document j?
- How much does each topic “like” the word Brexit?
- **Sample from the revised conditional distribution** $p(z_{ji} = k \,|\, \mathbf{z}_{j,-i}, \mathbf{W}, \boldsymbol{\alpha}, \boldsymbol{\eta})$
- …

document j

$\mathbf{z_{ji}}$	?	2	3	1	1
$\mathbf{w_{ji}}$	Brexit	deficit	Europe	market	single

words / topics	1	2	3
Brexit	100	30	1
deficit	10	60	0
Europe	95	5	2
market	50	70	5
single	50	15	90
…	…	…	…



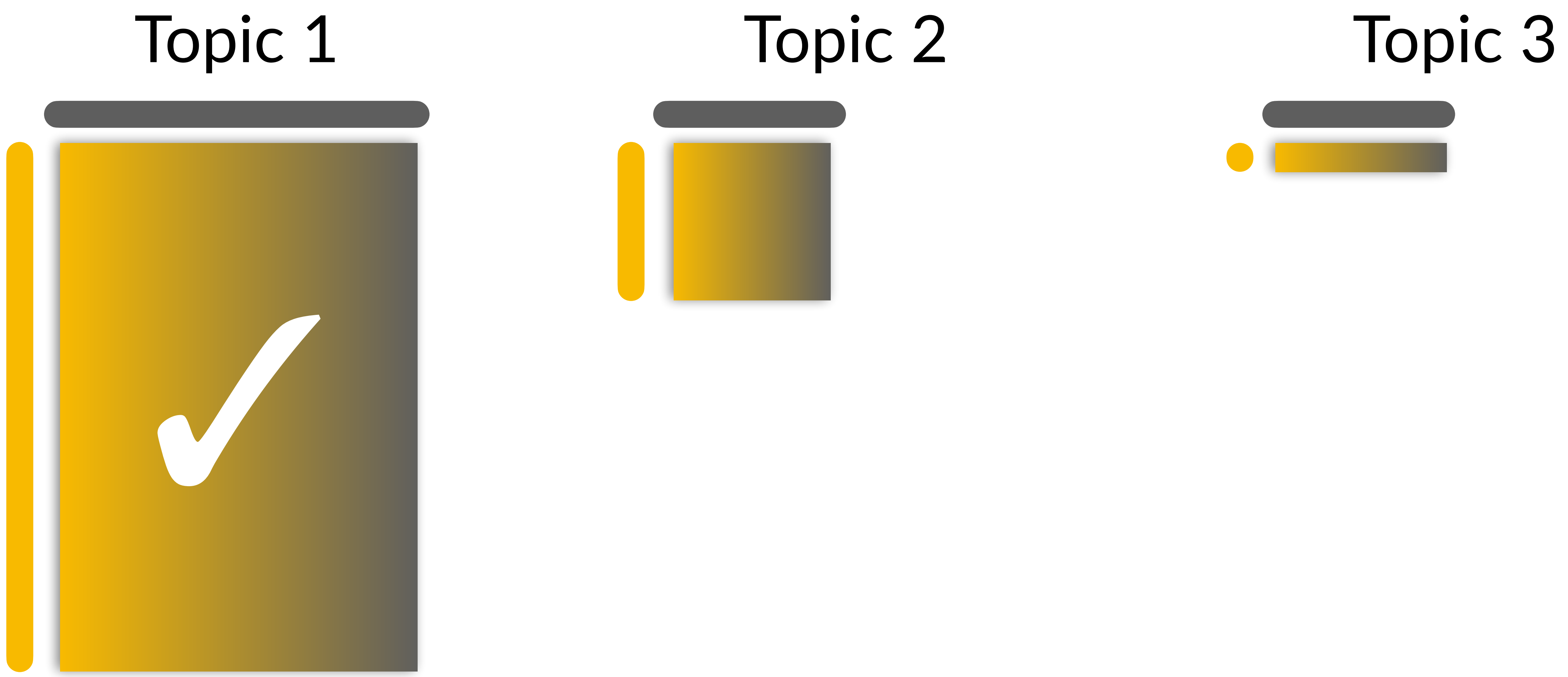
LDA — Gibbs sampling, toy example

- …
- What are the revised topic proportions in document *j*?
- How much does each topic “like” the word Brexit?
- Sample from the revised conditional distribution $p(z_{ji} = k \mid \mathbf{z}_{j,-i}, \mathbf{W}, \boldsymbol{\alpha}, \boldsymbol{\eta})$
- **Assign the sampled topic to the word “Brexit” and update counts**
- …

document *j*

z_{ji}	1	2	3	1	1
w_{ji}	Brexit	deficit	Europe	market	single

words / topics	1	2	3
Brexit	101	30	1
deficit	10	60	0
Europe	95	5	2
market	50	70	5
single	50	15	90
…	…	…	…



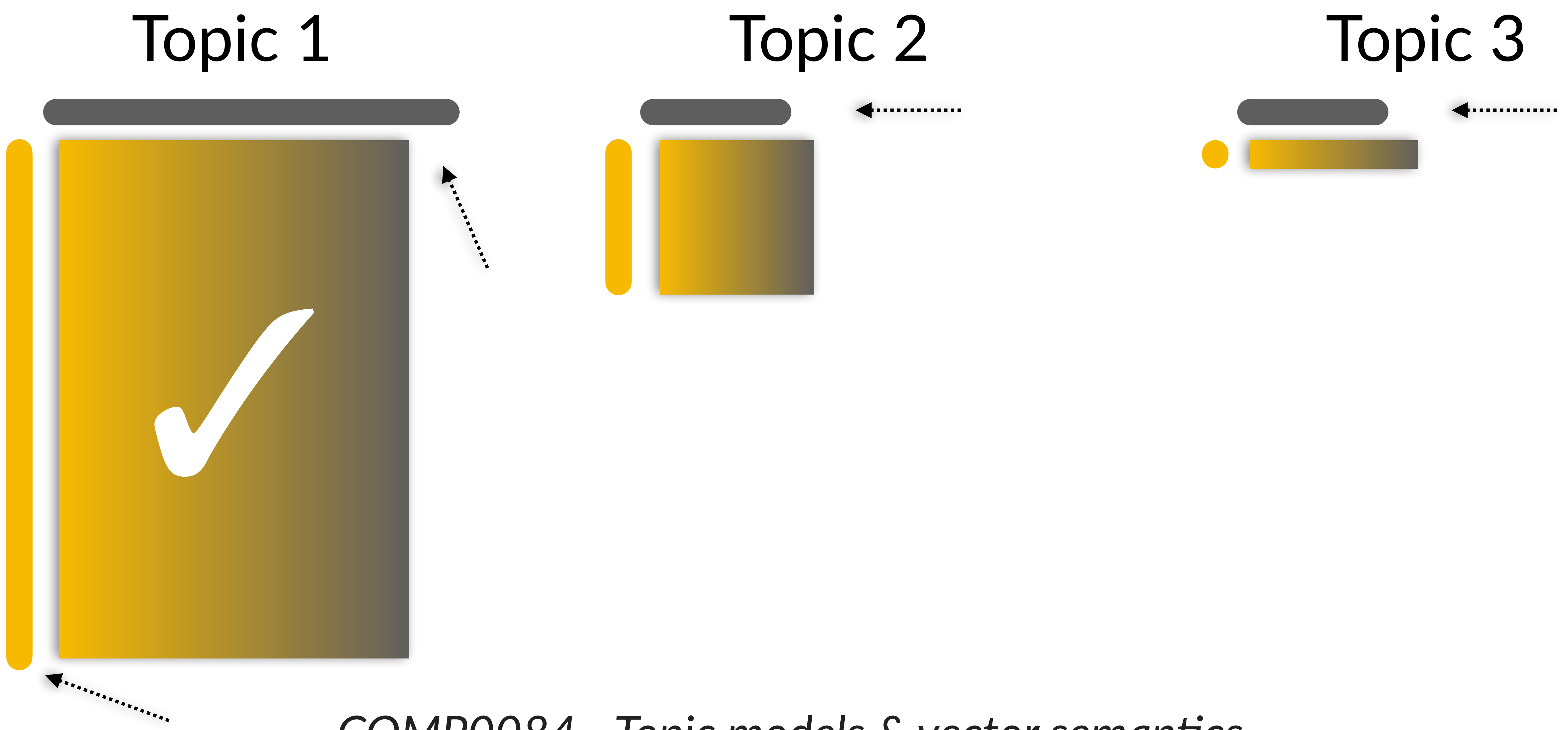
LDA — Gibbs sampling, toy example

- …
- What are the revised topic proportions in document *j*?
- How much does each topic "like" the word Brexit?
- Sample from the revised conditional distribution $p(z_{ji} = k \mid \mathbf{z}_{j,-i}, \mathbf{W}, \boldsymbol{\alpha}, \boldsymbol{\eta})$
- **Assign the sampled topic to the word "Brexit" and update counts**
- …

document *j*

z_{ji}	1	2	3	1	1
w_{ji}	Brexit	deficit	Europe	market	single

words / topics	1	2	3
Brexit	101	30	1
deficit	10	60	0
Europe	95	5	2
market	50	70	5
single	50	15	90
…	…	…	…



LDA online demo

mimno.infosci.cornell.edu/jsLDA/jslda.html

Evaluating topics

- **It depends** on what the topics are for!
- If they are generated for an end task with a measure-able performance, then we it makes sense to use this metric, *i.e.* the **performance of the end task** as a proxy for the value of the topic (<u>Note</u>: LDA tends to underperform in such settings)
- Compute the **probability of generating held-out documents** (*the higher the better*)
- **Word intrusion**: Show words from topics to human judges (*crowdsourcing*) with out-of-topic words inserted (intruders). How often can they identify the word that does not belong?

Words as vectors

- We've seen that documents can be represented as vectors of word frequencies
- **Words** can also be represented as multi-dimensional **vectors**
- **Property** to exploit: words that occur in similar contexts (co-occur) tend to have similar meanings

> **"You shall know a word by the company it keeps"**
> John Rupert (J. R.) Firth (1957)

Words as vectors

- We’ve seen that documents can be represented as vectors of word frequencies
- **Words** can also be represented as multi-dimensional **vectors**
- **Property** to exploit: words that occur in similar contexts (co-occur) tend to have similar meanings

“**You shall know a word by the company it keeps**”
John Rupert (J. R.) Firth (1957)

— My new **W** is much thinner than my previous one.
— I prefer to work from remote locations using a **W**.
— This old **W** has less RAM than my new smartphone.
— With a 15-inch display, it’s not a **W** anymore!

Words as vectors

- **Property** to exploit: words that occur in similar contexts (co-occur) tend to have similar meanings

  — My new **W** is much thinner than my previous one.
  — I prefer to work from remote locations using a **W**.
  — This old **W** has less RAM than my new smartphone.
  — With a 15-inch display, it's not a **W** anymore!

- Co-occurs with: "my", "thinner", "remote", "smartphone", "RAM", "display"
- Occurs after: "my", "a", "new", "old", "display"
- Occurs before: "has", "RAM", "thinner"
- **W** = **???**

Words as vectors

- **Property** to exploit: words that occur in similar contexts (co-occur) tend to have similar meanings

  — My new **W** is much thinner than my previous one.
  — I prefer to work from remote locations using a **W**.
  — This old **W** has less RAM than my new smartphone.
  — With a 15-inch display, it's not a **W** anymore!

- Co-occurs with: "my", "thinner", "remote", "smartphone", "RAM", "display"
- Occurs after: "my", "a", "new", "old", "display"
- Occurs before: "has", "RAM", "thinner"
- **W** = **laptop** / **notebook** / **tablet**

Using word context: Words as vectors

- Generate a **word-word** matrix
  — a.k.a. **word-context** or **word co-occurrence** matrix
  — Note: words can be "terms" in practice

Using word context: Words as vectors

- Generate a **word-word** matrix
  — a.k.a. **word-context** or **word co-occurrence** matrix
  — Note: words can be "terms" in practice
- If the size of our vocabulary is V, then the size of this matrix is commonly $V \times V$
- Each **cell** of the matrix reports a count of how many times two terms **co-occur** within a predefined context

Using word context: Words as vectors

- Generate a **word-word** matrix
  – a.k.a. **word-context** or **word co-occurrence** matrix
  – Note: words can be “terms” in practice
- If the size of our vocabulary is V, then the size of this matrix is commonly $V \times V$
- Each **cell** of the matrix reports a count of how many times two terms **co-occur** within a predefined context
- Possible **contexts**: entire document, a paragraph in a document, a sentence, a number of terms (window, commonly ± 4 words)

context **target words** **context**

… more succinct definition of **computer** science is the study…
… analysis and study of **algorithms**, discipline of computer science…
… the arrival of Japanese **mandarin** oranges signalled the real…
… of pomelo and mandarin, **orange** has genes from both…

Using word context: Words as vectors

… more succinct definition of **computer** science is the study…
… analysis and study of **algorithms**, discipline of computer science…
… the arrival of Japanese **mandarin** oranges signalled the real…
… of pomelo and mandarin, **orange** has genes from both…

word-word (word co-occurrence) matrix

target words

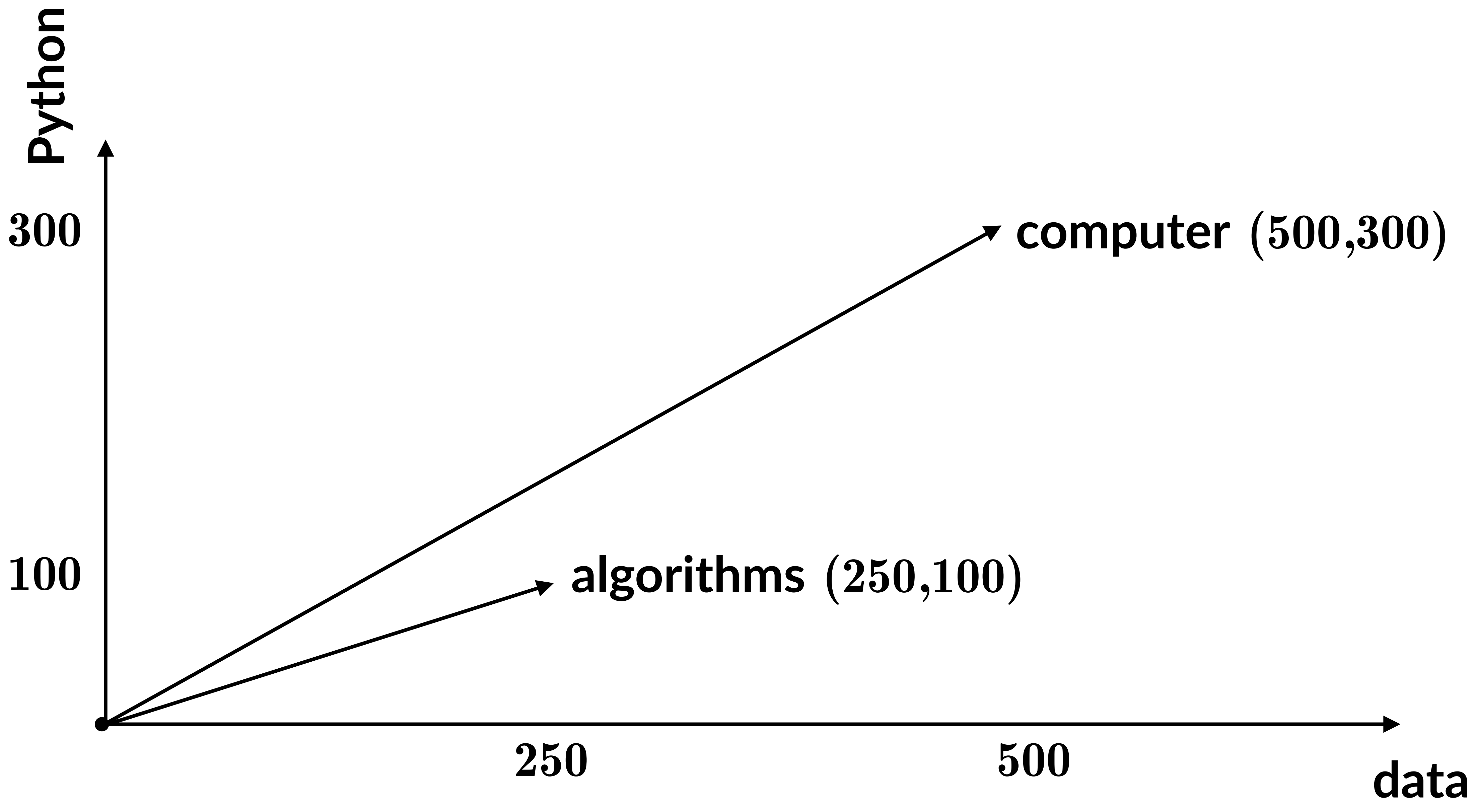
	…	data	…	fruit	…	Python	…
…	…	…	…	…	…	…	…
algorithms	…	100	…	2	…	250	…
…	…	…	…	…	…	…	…
computer	…	300	…	5	…	200	…
…	…	…	…	…	…	…	…
mandarin	…	1	…	300	…	0	…
…	…	…	…	…	…	…	…
orange	…	1	…	256	…	10	…
…	…	…	…	…	…	…	…

context words

Using word context: Words as vectors

	...	data	...	fruit	...	Python	...
...	…	…	…	…	…	…	…
algorithms	…	100	…	2	…	250	…
...	…	…	…	…	…	…	…
computer	…	300	…	5	…	200	…
...	…	…	…	…	…	…	…
mandarin	…	1	…	300	…	0	…
...	…	…	…	…	…	…	…
orange	…	1	…	256	…	10	…
...	…	…	…	…	…	…	…

We can use the word-context matrix to project words into space



Words as large, sparse vectors

- Recap: Word-context matrix of size $V \times V$ where V is the size of the vocabulary
- **Large** matrix as V is often very large ($>100{,}000$ terms)
- **Sparse** matrix as many entries will be 0 (not all words co-occur in all contexts)

Words as large, sparse vectors

- Recap: Word-context matrix of size $V \times V$ where V is the size of the vocabulary
- **Large** matrix as V is often very large ($>$100,000 terms)
- **Sparse** matrix as many entries will be 0 (not all words co-occur in all contexts)
- Small context window: a more **syntactic** representation (driven by syntax, grammar)
- Longer context window: a more **semantic** representation (more abstract connections may be captured)

Measuring word association — Pointwise Mutual Information (PMI)

- Raw word counts are not the best measure for word association — skewed towards frequent/infrequent words, non discriminative
- **Pointwise Mutual Information** (**PMI**) is a measure of how often two events co-occur, compared to what we would expect if these events were independent

Measuring word association — Pointwise Mutual Information (PMI)

- Raw word counts are not the best measure for word association — skewed towards frequent/infrequent words, non discriminative

- **Pointwise Mutual Information** (**PMI**) is a measure of how often two events co-occur, compared to what we would expect if these events were independent

- Centre (target) word w_i, context word c_j

$$\text{PMI}(w_i, c_j) = \log_2 \frac{p(w_i, c_j)}{p(w_i) \cdot p(c_j)}$$

- **Numerator:** How often we have seen these words together

- **Denominator:** How often we expect the words to co-occur, assuming they are independent

- **PMI:** how much more w_i, c_j co-occur than expected by chance

Positive Pointwise Mutual Information (PPMI)

- PMI ranges in $(-\infty, +\infty)$
- Negative PMI values are harder to interpret and evaluate
  — “relatedness” is easier to evaluate as opposed to “un-relatedness”
- Force positivity — Positive PMI (PPMI)

$$\text{PPMI}(w_i, c_j) = \max\left(\log_2 \frac{p(w_i, c_j)}{p(w_i) \cdot p(c_j)}, 0\right)$$

Computing PPMI

Assume a word-context matrix $\mathbf{A}$ of size $V \times C$; generalisation of the word-word matrix, where the C contexts may not be identical to the V target words. Let's generate a PPMI matrix from that.

$$\mathrm{PPMI}(w_i, c_j) = \max\left(\log_2 \frac{p(w_i, c_j)}{p(w_i) \cdot p(c_j)}, 0\right)$$

Computing PPMI

Assume a word-context matrix $\mathbf{A}$ of size $V \times C$; generalisation of the word-word matrix, where the C contexts may not be identical to the V target words. Let's generate a PPMI matrix from that.

$$\text{PPMI}(w_i, c_j) = \max\left(\log_2 \frac{p(w_i, c_j)}{p(w_i) \cdot p(c_j)}, 0\right)$$

$$p(w_i, c_j) = \frac{n_{ij}}{\sum_{i=1}^{V}\left(\sum_{j=1}^{C} n_{ij}\right)}$$

# target word w_i co-occurs with context word c_j divided by the total count of word occurrences in the corpus

$$p(w_i) = \frac{\sum_{j=1}^{C} n_{ij}}{\sum_{i=1}^{V}\left(\sum_{j=1}^{C} n_{ij}\right)}$$

# target word w_i appears in the corpus (sum of row i of $\mathbf{A}$) divided by…

$$p(c_j) = \frac{\sum_{i=1}^{V} n_{ij}}{\sum_{i=1}^{V}\left(\sum_{j=1}^{C} n_{ij}\right)}$$

# context word c_j appears in the corpus (sum of col. j of $\mathbf{A}$) divided by…

Computing PPMI

Assume a word-context matrix $\mathbf{A}$ of size $V \times C$; generalisation of the word-word matrix, where the C contexts may not be identical to the V target words. Let's generate a PPMI matrix from that.

$$\text{PPMI}(w_i, c_j) = \max\left(\log_2 \frac{p(w_i, c_j)}{p(w_i) \cdot p(c_j)}, 0\right)$$

Let's use the PPMI matrix to measure how (semantically) similar different words are. We will need a **similarity metric** for that.

$$p(w_i, c_j) = \frac{n_{ij}}{\sum_{i=1}^{V}\left(\sum_{j=1}^{C} n_{ij}\right)}$$

# target wo

the total co

$$p(w_i) = \frac{\sum_{j=1}^{C} n_{ij}}{\sum_{i=1}^{V}\left(\sum_{j=1}^{C} n_{ij}\right)}$$

# target word w_i appears in the corpus (sum of row i of $\mathbf{A}$) divided by…

$$p(c_j) = \frac{\sum_{i=1}^{V} n_{ij}}{\sum_{i=1}^{V}\left(\sum_{j=1}^{C} n_{ij}\right)}$$

# context word c_j appears in the corpus (sum of col. j of $\mathbf{A}$) divided by…

Measuring word similarity — Cosine

- Dot product between word vectors $\mathbf{w}$, $\mathbf{v}$: $\mathbf{w}^\top \mathbf{v} = \sum_{i=1}^{N} w_i \cdot v_i$

  — **Not balanced:** Greater values for longer vectors and for frequent words

Measuring word similarity — Cosine

- Dot product between word vectors $\mathbf{w}$, $\mathbf{v}$: $\mathbf{w}^\top \mathbf{v} = \sum_{i=1}^{N} w_i \cdot v_i$

  — **Not balanced:** Greater values for longer vectors and for frequent words

- Normalise it by dividing with the length of the vectors! This leads to cosine similarity, *i.e.* the cosine of the angle (ϕ) between the two vectors

$$\text{cosine-sim}(\mathbf{w}, \mathbf{v}) = \frac{\sum_{i=1}^{N} w_i \cdot v_i}{\sqrt{\sum_{i=1}^{N} w_i^2} \cdot \sqrt{\sum_{i=1}^{N} v_i^2}} = \frac{\mathbf{w}^\top \mathbf{v}}{\|\mathbf{w}\|_2 \|\mathbf{v}\|_2} = \cos \phi$$

Measuring word similarity — Cosine

- Dot product between word vectors $\mathbf{w}$, $\mathbf{v}$: $\mathbf{w}^\top \mathbf{v} = \sum_{i=1}^{N} w_i \cdot v_i$

  – **Not balanced:** Greater values for longer vectors and for frequent words

- Normalise it by dividing with the length of the vectors! This leads to cosine similarity, *i.e.* the cosine of the angle (ϕ) between the two vectors

$$\text{cosine-sim}(\mathbf{w}, \mathbf{v}) = \frac{\sum_{i=1}^{N} w_i \cdot v_i}{\sqrt{\sum_{i=1}^{N} w_i^2} \cdot \sqrt{\sum_{i=1}^{N} v_i^2}} = \frac{\mathbf{w}^\top \mathbf{v}}{\|\mathbf{w}\|_2 \|\mathbf{v}\|_2} = \cos \phi$$

- Since $\mathbf{w}$ and $\mathbf{v} > 0$ (***when using PPMI***), cosine-sim $(\mathbf{w}, \mathbf{v})$ ranges from $\left[0,1\right]$

  – cosine-sim $(\mathbf{w}, \mathbf{v}) = 0$ means that $\phi = 90°$

  – cosine-sim $(\mathbf{w}, \mathbf{v}) = 1$ means that $\phi = 0°$

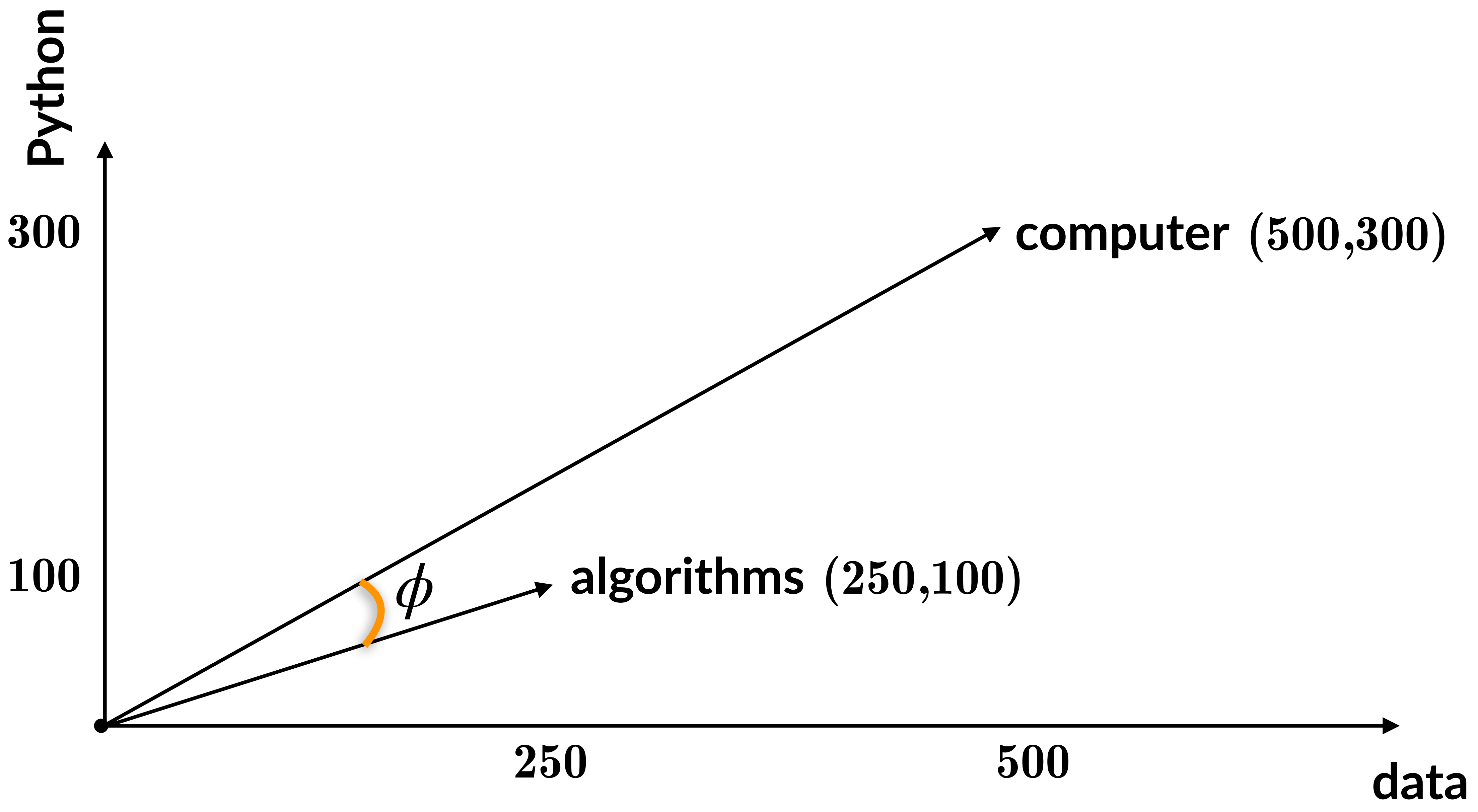
Measuring word similarity — Cosine

$$\text{cosine-sim}(\mathbf{w},\mathbf{v}) = \frac{\sum_{i=1}^{N} w_i \cdot v_i}{\sqrt{\sum_{i=1}^{N} w_i^2} \cdot \sqrt{\sum_{i=1}^{N} v_i^2}} = \frac{\mathbf{w}^\top \mathbf{v}}{\|\mathbf{w}\|_2 \|\mathbf{v}\|_2} = \cos\phi$$

- Since $\mathbf{w}$ and $\mathbf{v} > 0$ (***when using PPMI***), cosine-sim $(\mathbf{w},\mathbf{v})$ ranges from $[0,1]$

  – cosine-sim $(\mathbf{w},\mathbf{v}) = 0$ means that $\phi = 90°$

  – cosine-sim $(\mathbf{w},\mathbf{v}) = 1$ means that $\phi = 0°$



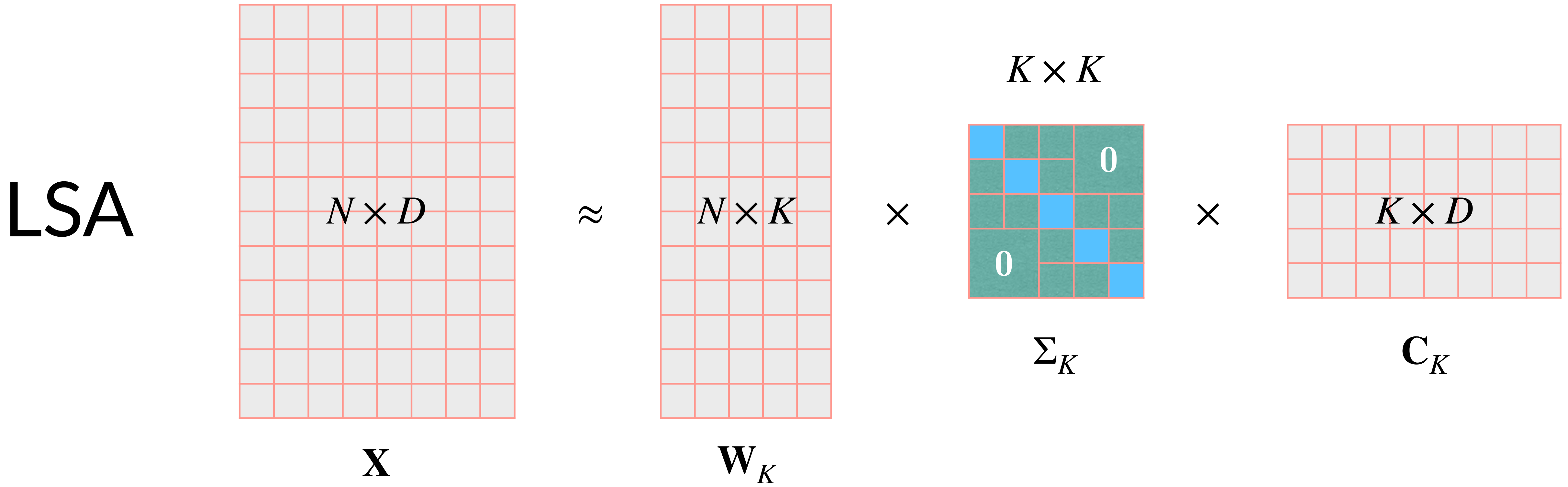
cosine-sim(computer, algorithms) $= 0.9872$

$\phi = 9.162°$

From sparse to dense word vectors

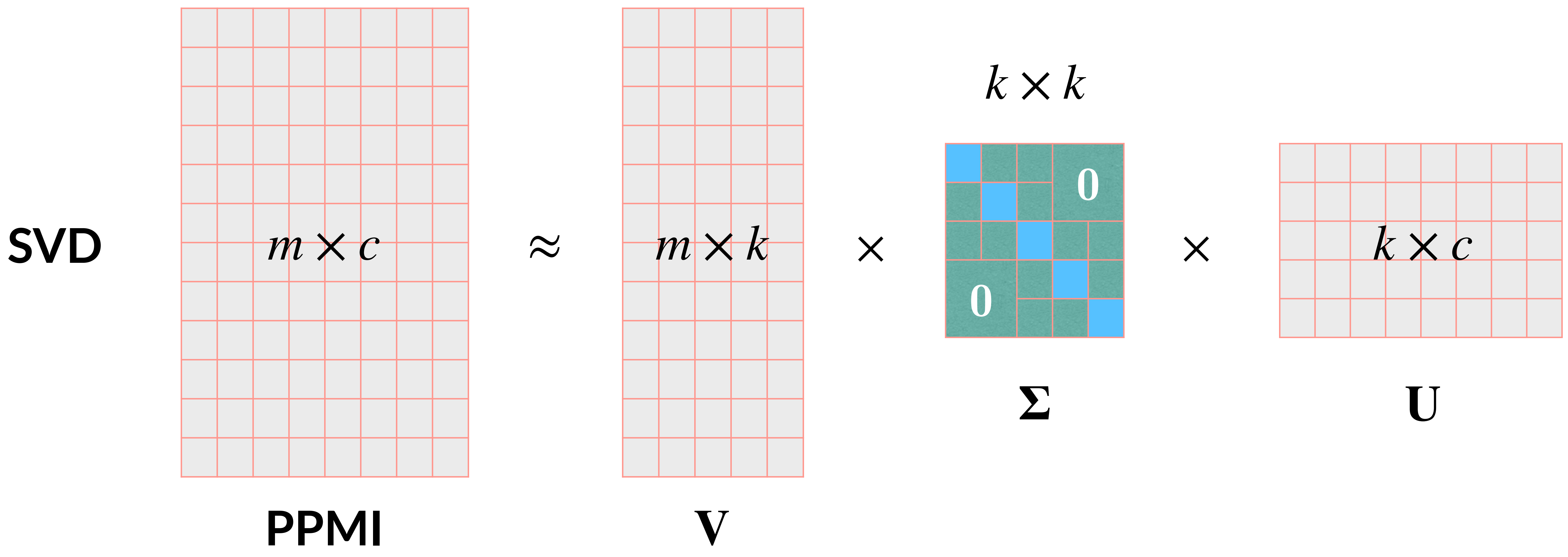
- Previously shown word representations: **long** (equal to size of the vocabulary V) and **sparse** (many 0's)

- **Short** and **dense** representations have advantages
  — easier to use as features in statistical learning methods
  — capture synonymy better
  — generalise better

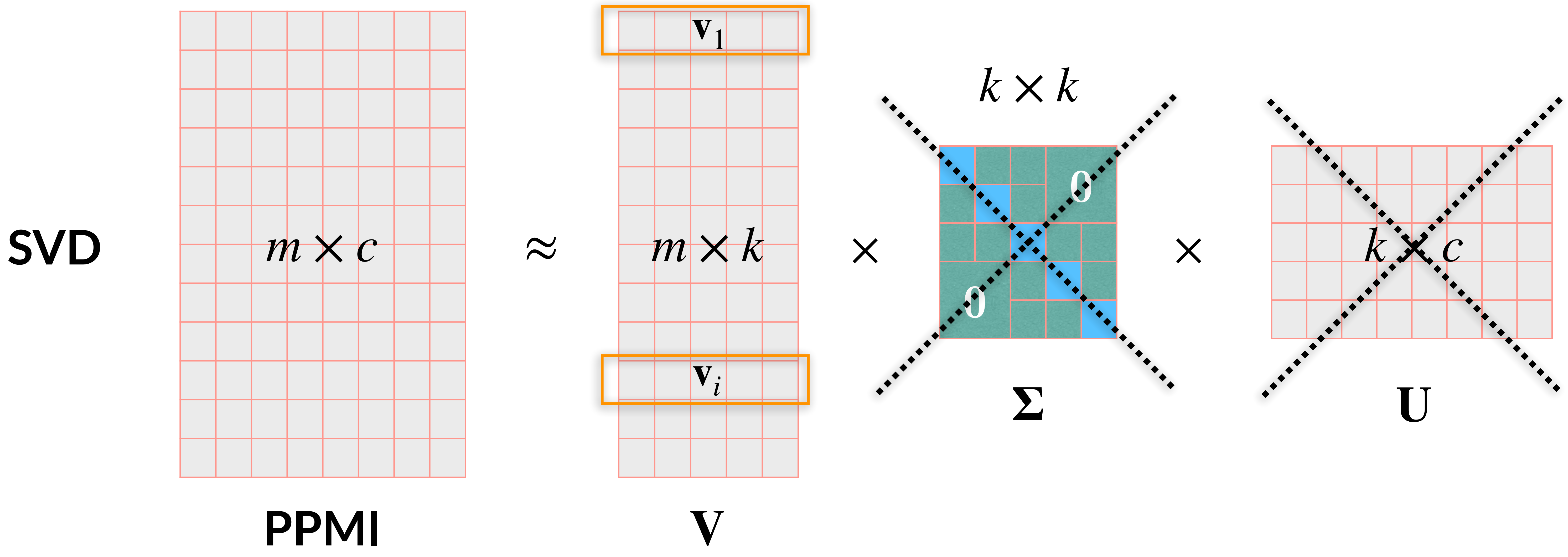


- Recall Latent Semantic Analysis (LSA), *i.e.* SVD on the word-document matrix, $\mathbf{X}$. What if we perform SVD on a word co-occurrence or a PPMI matrix?

SVD on the PPMI word-context matrix



SVD on the PPMI word-context matrix



- $\mathbf{v}_i$: k-dimensional vector that represents word i in our vocabulary
  – also known as a **word embedding**
  – commonly, $k = 128$ to 1024, i.e. $\mathbf{v}_i$ is short and dense

- **Downside:** SVD has a significant computational cost

Word embeddings from prediction

- Same intuition, *different* approach
  — words with similar meanings will co-occur
  — instead of counting co-occurrences, **predict** them
- First broadly adopted method: **word2vec** — title of the software library, but there is a small family of methods behind it
- Algorithms
  — ***skip-gram***: Predict the context (surrounding) words based on a centre word
  — CBOW (continuous bag-of-words): Predict a centre word based on the context words
- Training methods
  — Hierarchical softmax
  — Negative sampling
  — ***Naïve softmax***

word2vec — skip-gram

… said that “Hey Jude” is Beatles’ most famous song, but…

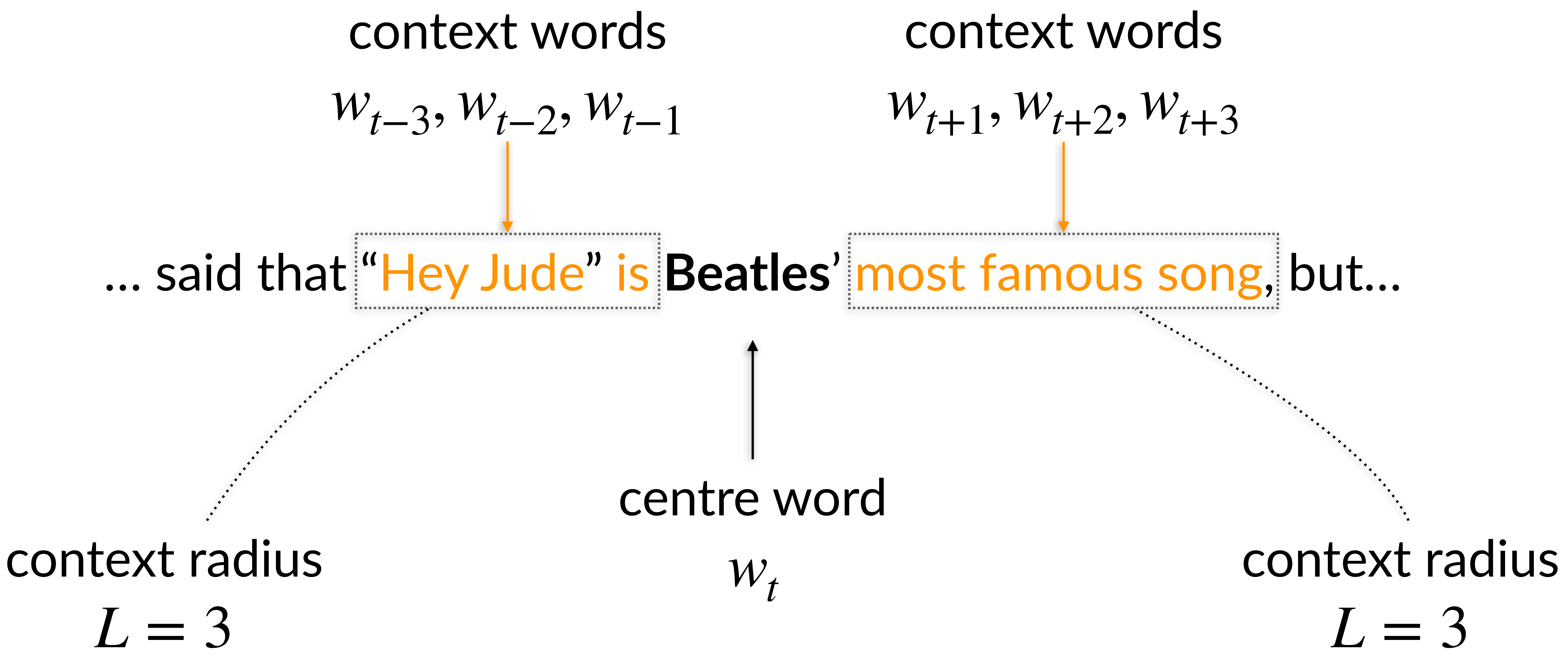
word2vec — skip-gram

… said that “Hey Jude” is **Beatles**’ most famous song, but…

centre word

w_t

word2vec — skip-gram



context words

$w_{t-3}, w_{t-2}, w_{t-1}$

context words

$w_{t+1}, w_{t+2}, w_{t+3}$

… said that "Hey Jude" is **Beatles**' most famous song, but…

context radius

$L = 3$

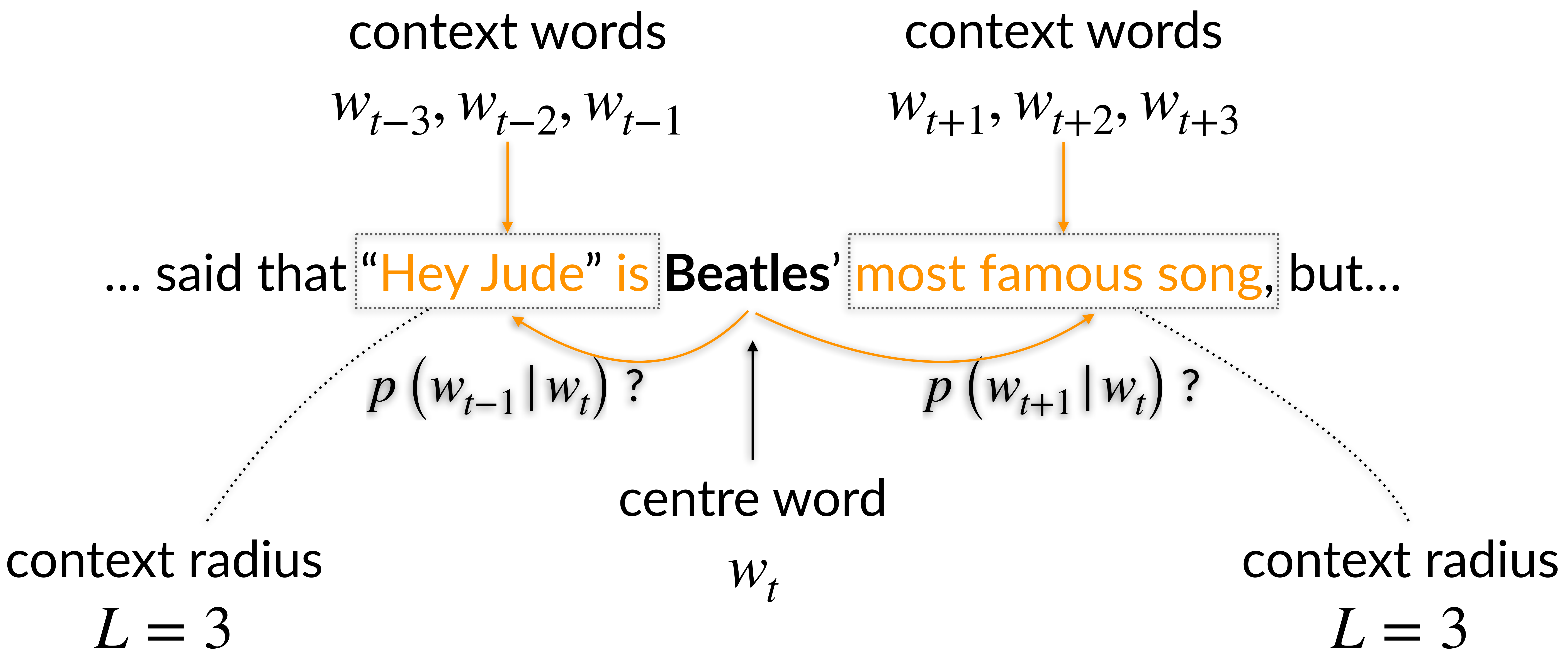
centre word

w_t

context radius

$L = 3$

word2vec — skip-gram



context words

$w_{t-3}, w_{t-2}, w_{t-1}$

context words

$w_{t+1}, w_{t+2}, w_{t+3}$

… said that “Hey Jude” is **Beatles**’ most famous song, but…

$p\left(w_{t-1} | w_t\right)$?

$p\left(w_{t+1} | w_t\right)$?

centre word

w_t

context radius

$L = 3$

context radius

$L = 3$

skip-gram — Simplified objective function

For each word position t out of T, predict the context words using a fixed radius L (or a symmetric window $2L$)

Objective: Maximise the probability of any context word given the current centre word (the position of surrounding words does not matter)

$$\max \prod_{t=1}^{T} \prod_{i=-L,\ i\neq 0}^{L} p\left(w_{t+i} \mid w_t\right)$$

skip-gram — Simplified objective function

For each word position t out of T, predict the context words using a fixed radius L (or a symmetric window $2L$)

Objective: Maximise the probability of any context word given the current centre word (the position of surrounding words does not matter)

$$\max \prod_{t=1}^{T} \prod_{i=-L,\ i\neq 0}^{L} p\left(w_{t+i} \,|\, w_t\right)$$

Prefer to minimise things, and sums over products

Minimise the mean (across all T samples) negative log likelihood:

$$\min \frac{1}{T}\left(-\sum_{t=1}^{T} \sum_{i=-L,\ i\neq 0}^{L} \log\left(p\left(w_{t+i} \,|\, w_t\right)\right)\right)$$

skip-gram — Simplified objective function

For each word position t out of T, predict the context words using a fixed radius L (or a symmetric window $2L$)

Objective: Maximise the probability of any context word given the current centre word

$$\min \frac{1}{T}\left(-\sum_{t=1}^{T}\sum_{i=-L,\ i\neq 0}^{L}\log\left(p\left(w_{t+i}|w_t\right)\right)\right)$$

- Assume that each centre word (t) has a k-dimensional (common setting for $k \in [100{,}1000]$) vector representation $\mathbf{v}_c$; all the vectors of the m centre words are held in an $k \times m$ matrix $\mathbf{V}$
- Assume that each context word has a k-dimensional vector representation $\mathbf{u}_x$; all the vectors of the m context words are held in an $k \times m$ matrix $\mathbf{U}$
- Thus, the model parameters ($2mk$) are now $\mathbf{Q} = \left[\mathbf{V}\ \mathbf{U}\right]$

$$\min_{\mathbf{Q}} \frac{1}{T}\left(-\sum_{t=1}^{T}\sum_{i=-L,\ i\neq 0}^{L}\log\left(p\left(w_{t+i}|w_t\,;\mathbf{Q}\right)\right)\right)$$

skip-gram — Simplified objective function

$$\min_{\mathbf{Q}} \frac{1}{T} \left(-\sum_{t=1}^{T} \sum_{i=-L,\ i\neq 0}^{L} \log \left(p \left(w_{t+i} \,|\, w_t \,; \mathbf{Q} \right) \right) \right)$$

We need an estimate of the probability $p(w_{t+1} \,|\, w_t)$ to insert into the formula above

To estimate this we will use a (*bad*) measure of similarity (dot product) and normalise it using a common approach in neural networks, the **softmax** function that converts a vector into a pseudo-probability distribution

Assuming a vocabulary of m words, for a centre word c ($\mathbf{v}_c$) and a context word x ($\mathbf{u}_x$)

$$p(x \,|\, c) = \frac{\exp\left(\mathbf{u}_x^\top \mathbf{v}_c\right)}{\sum_{w=1}^{m} \exp\left(\mathbf{u}_w^\top \mathbf{v}_c\right)}$$

skip-gram — In practice

$$w_t = \left[0\,0\ \ldots\ 1\ \ldots\ 0\right]^\top$$

centre word as an one-hot vector

$$v_c = \mathbf{V} \cdot w_t$$

get its vector representation (embedding) from the matrix of centre word embeddings

$$o = \mathbf{U}^\top \cdot v_c$$

dot product with all context word vectors m (voc. size) $\times 1$

$$p_{w_i} = \text{softmax}(o)_i$$

compute the softmax of this vector — this is the probability of word i, we have $2L$ context words

e.g. p_w

$$\begin{bmatrix} 0.1 \\ 0.4 \\ 0.01 \\ 0.09 \\ 0.05 \\ 0.25 \\ 0.08 \\ 0.02 \end{bmatrix} \longrightarrow \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

But we also know the correct answer!
In this case, we need to improve our
embeddings ($\mathbf{V}$ and $\mathbf{U}$).

In neural nets: do error back-propagation.

skip-gram — Negative sampling

Naïve / inefficient way for parameter inference

$$J\left(\mathbf{Q}\right) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{i=-L,\ i\neq 0}^{L}\log\left(p\left(w_{t+i}\,|\,w_t\,;\mathbf{Q}\right)\right)$$

Gradient descent: $\mathbf{Q}_{p+1} = \mathbf{Q}_p - \gamma\,\nabla_{\mathbf{Q}}J\left(\mathbf{Q}_p\right)$

Too slow and computationally expensive. Recall, the denominator is too expensive to compute (for large vocabularies; m)

$$p(x\,|\,c) = \frac{\exp\left(\mathbf{u}_x^\top\mathbf{v}_c\right)}{\sum_{w=1}^{m}\exp\left(\mathbf{u}_w^\top\mathbf{v}_c\right)}$$

skip-gram — Negative sampling

Naïve / inefficient way for parameter inference

$$J\left(\mathbf{Q}\right) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{i=-L,\ i\neq 0}^{L} \log\left(p\left(w_{t+i}\,|\,w_t\,;\mathbf{Q}\right)\right)$$

Gradient descent: $\mathbf{Q}_{p+1} = \mathbf{Q}_p - \gamma\,\nabla_{\mathbf{Q}} J\left(\mathbf{Q}_p\right)$

Too slow and computationally expensive. Recall, the denominator is too expensive to compute (for large vocabularies; m)

$$p(x\,|\,c) = \frac{\exp\left(\mathbf{u}_x^\top \mathbf{v}_c\right)}{\sum_{w=1}^{m}\exp\left(\mathbf{u}_w^\top \mathbf{v}_c\right)}$$

Negative sampling: For each context word, sample non-neighbouring words as “negative” samples

New objective: High dot product with context words and low dot product with “negative” samples

skip-gram — Negative sampling

Naïve / inefficient way for parameter inference

$$J\left(\mathbf{Q}\right) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{i=-L,\ i\neq 0}^{L}\log\left(p\left(w_{t+i}\,|\,w_t\,;\mathbf{Q}\right)\right)$$

Gradient descent: $\mathbf{Q}_{p+1} = \mathbf{Q}_p - \gamma\,\nabla_{\mathbf{Q}}J\left(\mathbf{Q}_p\right)$

Going over all the training samples (for a gradient update) is also slow.

skip-gram — Negative sampling

Naïve / inefficient way for parameter inference

$$J(\mathbf{Q}) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{i=-L,\ i\neq 0}^{L} \log\Big(p\left(w_{t+i}\,|\,w_t\,;\mathbf{Q}\right)\Big)$$

Gradient descent: $\mathbf{Q}_{p+1} = \mathbf{Q}_p - \gamma\,\nabla_{\mathbf{Q}} J\left(\mathbf{Q}_p\right)$

Going over all the training samples (for a gradient update) is also slow.

Apply mini-batch gradient descent:

i.e. instead of going through all the data for computing $\nabla_{\mathbf{Q}} J\left(\mathbf{Q}_p\right)$

we use one or small subsets of the data (mini batches) to update the gradient

Word analogies with word embeddings

Note

Word embeddings tend to carry the biases or stereotypes of the corpora used to train them!

$$\underset{b}{\text{vector('}\textbf{\textit{queen}}\text{')}} \approx \underset{a}{\text{vector('}\textbf{\textit{king}}\text{')}} - \underset{a_p}{\text{vector('}\textbf{\textit{man}}\text{')}} + \underset{b_p}{\text{vector('}\textbf{\textit{woman}}\text{')}}$$

cosine similarity between 'queen' and 'king' - 'man' + 'woman'

$$b = \arg\max_{b \in V} \left(\cos \left(\mathbf{v}_b, \mathbf{v}_a - \mathbf{v}_{a_p} + \mathbf{v}_{b_p} \right) \right)$$

Compute the cosine similarity between the composite embedding $\left(\mathbf{v}_a - \mathbf{v}_{a_p} + \mathbf{v}_{b_p} \right)$ and each other embedding in our vocabulary, and expect that $\mathbf{v}_b =$ vector('***queen***') will have the greatest one.

This gives rise to the **word analogy**

a_p is for a, what b_p is for b

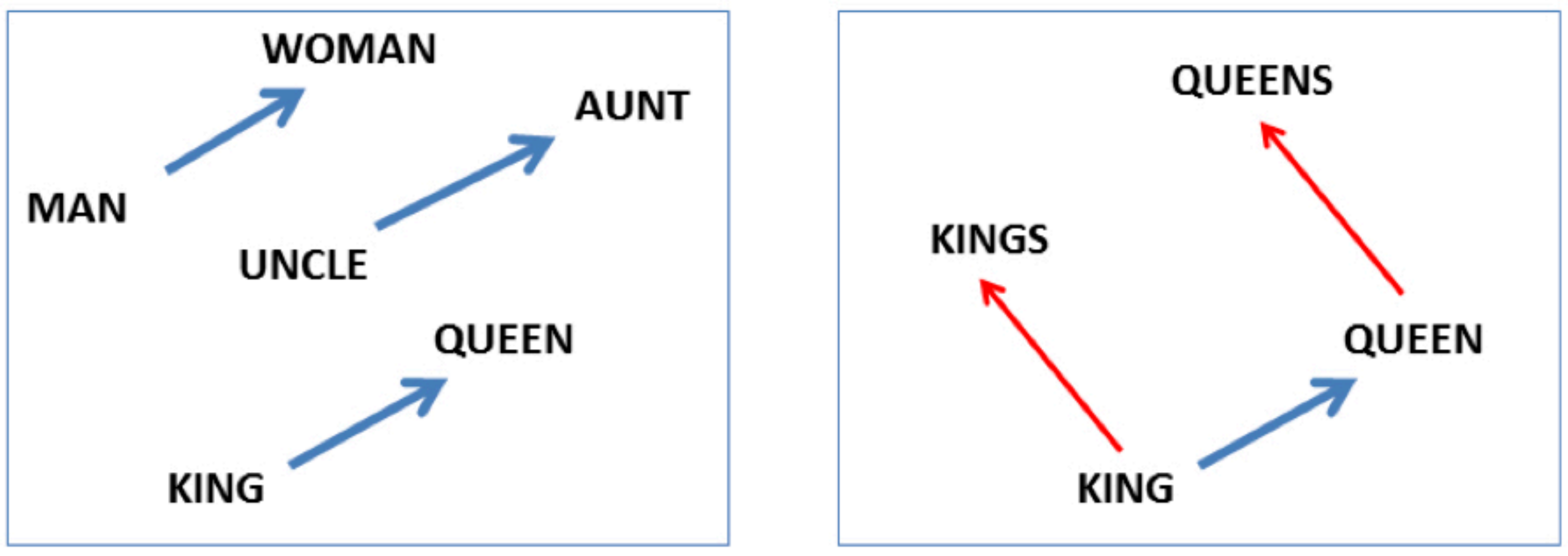
or *man* is for *king*, what *woman* is for *queen*

Word analogies with word embeddings

Note

Word embeddings tend to carry the biases or stereotypes of the corpora used to train them!

$$\underset{b}{\text{vector}(\text{‘}\textbf{\textit{queen}}\text{’})} \approx \underset{a}{\text{vector}(\text{‘}\textbf{\textit{king}}\text{’})} - \underset{a_p}{\text{vector}(\text{‘}\textbf{\textit{man}}\text{’})} + \underset{b_p}{\text{vector}(\text{‘}\textbf{\textit{woman}}\text{’})}$$



This gives rise to the **word analogy**

a_p is for a, what b_p is for b

or *man* is for *king*, what *woman* is for *queen*

Word embeddings based on Twitter

word2vec embeddings

- trained (*a few years back*) on 1.1 billion tweets post during 2012 to 2016, approximately geolocated in the UK
- tweets represent current trends, include informal forms of language, and are often topic-consistent
- $470{,}194$ terms covered (size of the vocabulary)
- the dimensionality of the embedding is equal to 512
- available online at figshare.com/articles/UK_Twitter_word_embeddings_II_/5791650

Twitter word embeddings — Similarities

Top-5 most similar words using cosine similarity on word embeddings

- **Monday:** Tuesday, Thursday, Wednesday, Friday, Sunday
- **January:** February, August, October, March, June
- **red:** yellow, blue, purple, pink, green
- **we:** they, you, we've, our, us
- **espresso:** expresso, cappuccino, macchiato, latte, coffee
- **linux:** Unix, Centos, Debian, Ubuntu, Redhat
- **retweet:** rt, tweet, retweets, retweeting, rewteet
- **democracy:** democratic, dictatorship, democracies, socialism, undemocratic
- **loool:** looool, lool, loooool, looooool, loooooool
- **xxxx:** xxxxx, xxx, xxxxxxxx, xxxxxx, xxxxxxx
- **enviroment:** environment, environments, env, enviro, habitats

Twitter word embeddings — Analogies

- **she** is to **her** what **he** is to …

Twitter word embeddings — Analogies

- **she** is to **her** what **he** is to ... [**his**, **him**, himself]

Twitter word embeddings — Analogies

- **she** is to **her** what **he** is to ... [**his**, **him**, himself]
- **Rome** is to **Italy** what **London** is to ...

Twitter word embeddings — Analogies

- **she** is to **her** what **he** is to ... [**his**, **him**, himself]
- **Rome** is to **Italy** what **London** is to ... [**UK**, Denmark, Sweden]

Twitter word embeddings — Analogies

- **she** is to **her** what **he** is to ... [**his**, **him**, himself]
- **Rome** is to **Italy** what **London** is to ... [**UK**, Denmark, Sweden]
- **go** is for **went** what **do** is to... [**did**, doing, happened]

Twitter word embeddings — Analogies

- **she** is to **her** what **he** is to ... [**his**, **him**, himself]
- **Rome** is to **Italy** what **London** is to ... [**UK**, Denmark, Sweden]
- **go** is for **went** what **do** is to... [**did**, doing, happened]
- **big** is to **bigger** what **small** is to... [**smaller**, larger, tiny]

Twitter word embeddings — Analogies

- **she** is to **her** what **he** is to ... [**his**, **him**, himself]
- **Rome** is to **Italy** what **London** is to ... [**UK**, Denmark, Sweden]
- **go** is for **went** what **do** is to... [**did**, doing, happened]
- **big** is to **bigger** what **small** is to... [**smaller**, larger, tiny]
- **poet** is to **poem** what **author** is to... [**novel**, excerpt, memoir]
- **Messi** is to **football** what **Lebron** is to... [**basketball**, bball, NBA]
- **Elvis** is to **Presley** what **Aretha** is to... [**Franklin**, Ruffin, Vandross]

Twitter word embeddings — Analogies

- **she** is to **her** what **he** is to ... [**his**, **him**, himself]
- **Rome** is to **Italy** what **London** is to ... [**UK**, Denmark, Sweden]
- **go** is for **went** what **do** is to... [**did**, doing, happened]
- **big** is to **bigger** what **small** is to... [**smaller**, larger, tiny]
- **poet** is to **poem** what **author** is to... [**novel**, excerpt, memoir]
- **Messi** is to **football** what **Lebron** is to... [**basketball**, bball, NBA]
- **Elvis** is to **Presley** what **Aretha** is to... [**Franklin**, Ruffin, Vandross]
- **UK** is for **Brexit** what **Greece** is to... [**Grexit**, Syriza, Tsipras]
- **UK** is for **Farage** what **USA** is to... [**Trump**, Farrage, Putin]

Next lectures with me

- March 20, 11am to 12pm, guest lecture by me about some of my research