

# Topic modeling

Hyunjoong Kim

[soy.lovit@gmail.com](mailto:soy.lovit@gmail.com)

[github.com/lovit](https://github.com/lovit)

Latent Semantic Indexing (LSI)

probabilistic Latent Semantic Indexing (pLSA)

Latent Dirichlet Allocation (LDA)

# Latent Semantic Indexing (LSI)

---

- Latent Semantic Analysis (LSA)라고도 불립니다.
- 단어 – 문서 행렬에 SVD 를 적용하여 단어와 문서를 토픽 공간의 벡터로 표현합니다.
  - $m \times n$  (단어, 문서) 행렬이
  - $m \times t, t \times n$  의 (단어, 토픽), (토픽, 문서) 행렬로 분해됩니다.

# Singular Value Decomposition

- SVD 는 행렬  $A$  를 다음처럼 세 개의 행렬로 분해합니다.

$$A = U \Sigma V^T$$

The diagram illustrates the Singular Value Decomposition (SVD) of matrix  $A$ . Matrix  $A$  (dimensions  $m \times n$ ) is decomposed into three matrices:  $U$  (dimensions  $m \times m$ ),  $\Sigma$  (dimensions  $m \times n$ ), and  $V^T$  (dimensions  $n \times n$ ).

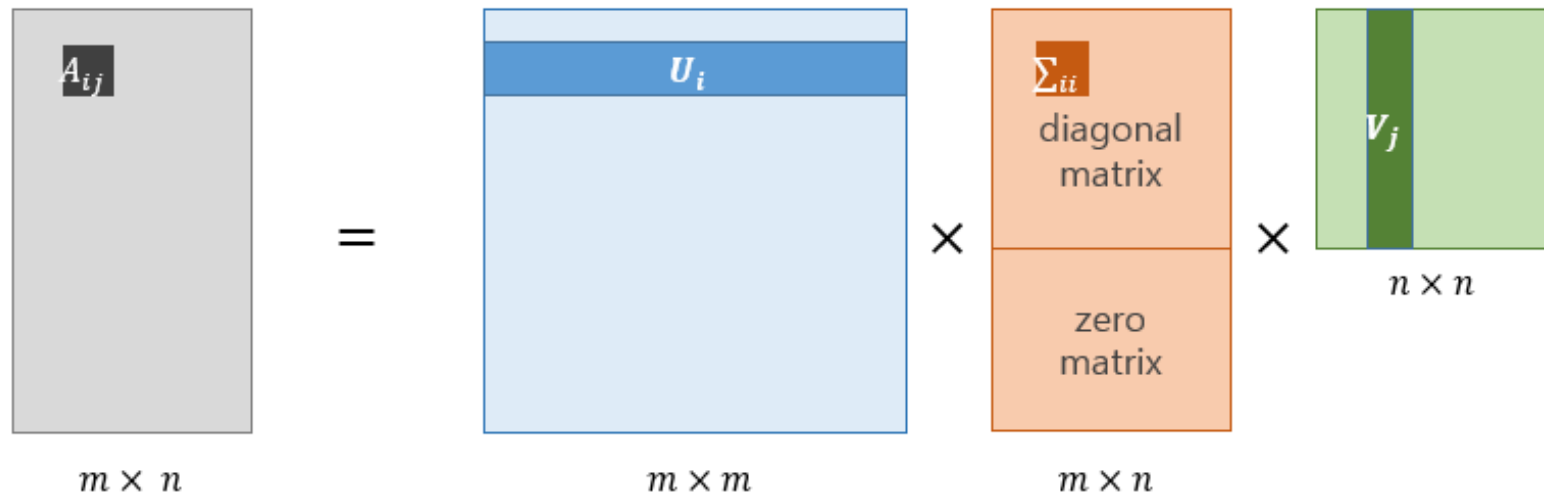
Matrix  $U$  is represented by a light blue square with a dark blue horizontal band at the top labeled  $U_i$ .

Matrix  $\Sigma$  is represented by an orange rectangle divided into two sections: the top section is labeled  $\Sigma_{ii}$  diagonal matrix, and the bottom section is labeled zero matrix.

Matrix  $V^T$  is represented by a light green rectangle with a dark green vertical band on the left labeled  $V_j$ .

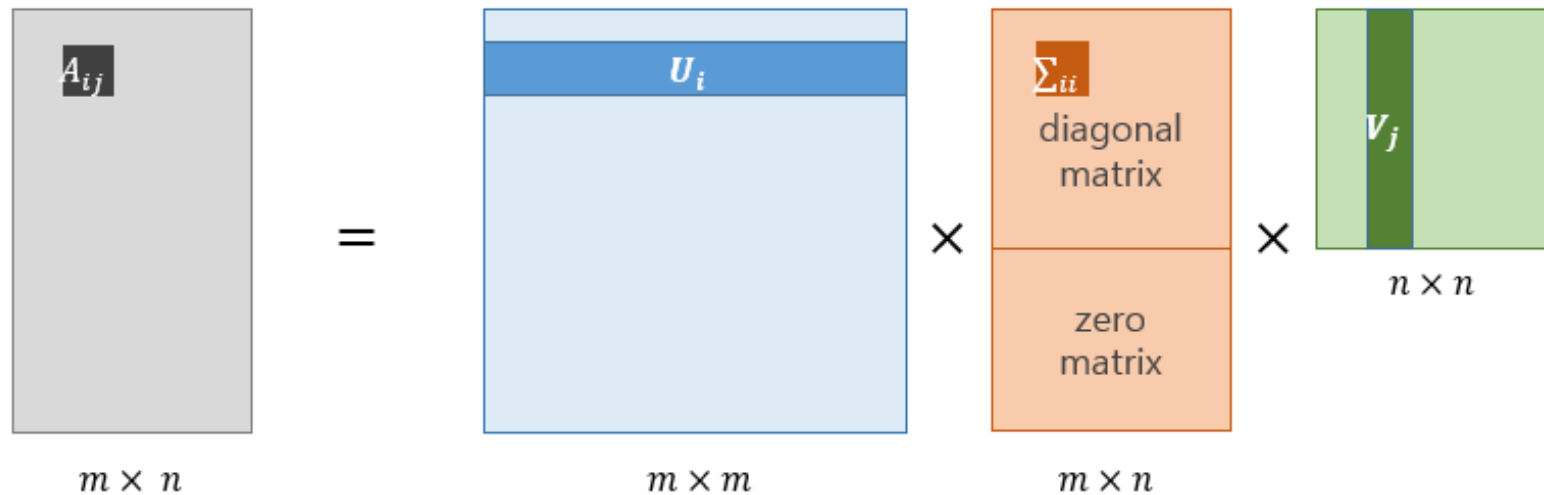
# Singular Value Decomposition

- $U, V$  는 각각 orthonormal matrix 입니다.
  - 각 행과 열의 벡터의 크기는 1 이며 (normal)
  - $U_i \times U_j^T = 0, U_i \cdot U_i^T = 1$  입니다.
  - $V$  도 동일합니다.



# Singular Value Decomposition

- $\Sigma$  는 PCA 에서의 각 component 별 중요도 (variance proportion) 입니다.
  - $\min(m, n)$  개의 components 에 대한 중요도로 해석할 수 있습니다.



# Singular Value Decomposition

- truncated SVD 는  $\Sigma$  에서 중요한  $k$  개의 components 만 이용합니다.
- $U, V$  를  $k$  차원으로 축소한 것으로 해석할 수 있습니다.

$$\begin{matrix} \boxed{A_{ij}} \\ m \times n \end{matrix} = \begin{matrix} \boxed{U_i} \\ m \times k \end{matrix} \times \begin{matrix} \boxed{\Sigma_{ii}} \\ k \times k \end{matrix} \times \begin{matrix} \boxed{V_j} \\ k \times n \end{matrix}$$

# Singular Value Decomposition

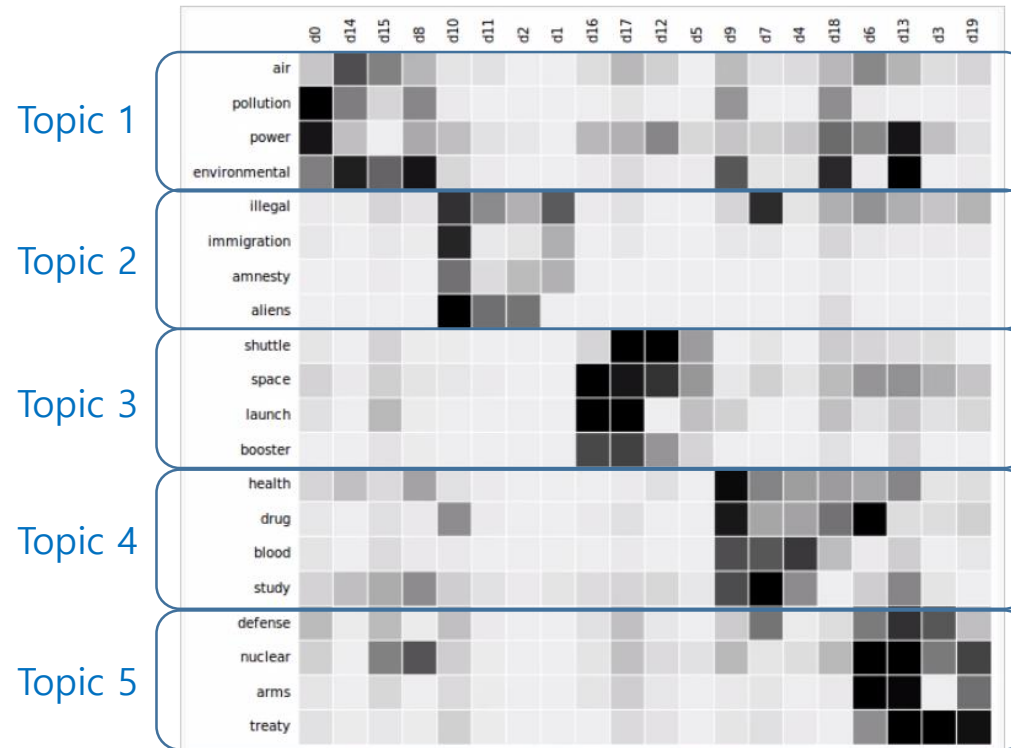
---

- Top principal components 에서 중요한 정보들은 추출이 되었기 때문에 정보의 손실이 적습니다.
- 비슷한 문서들에서 함께 등장한 단어들의 정보가 각 components 에 저장됩니다.



# Latent Semantic Indexing (LSI)

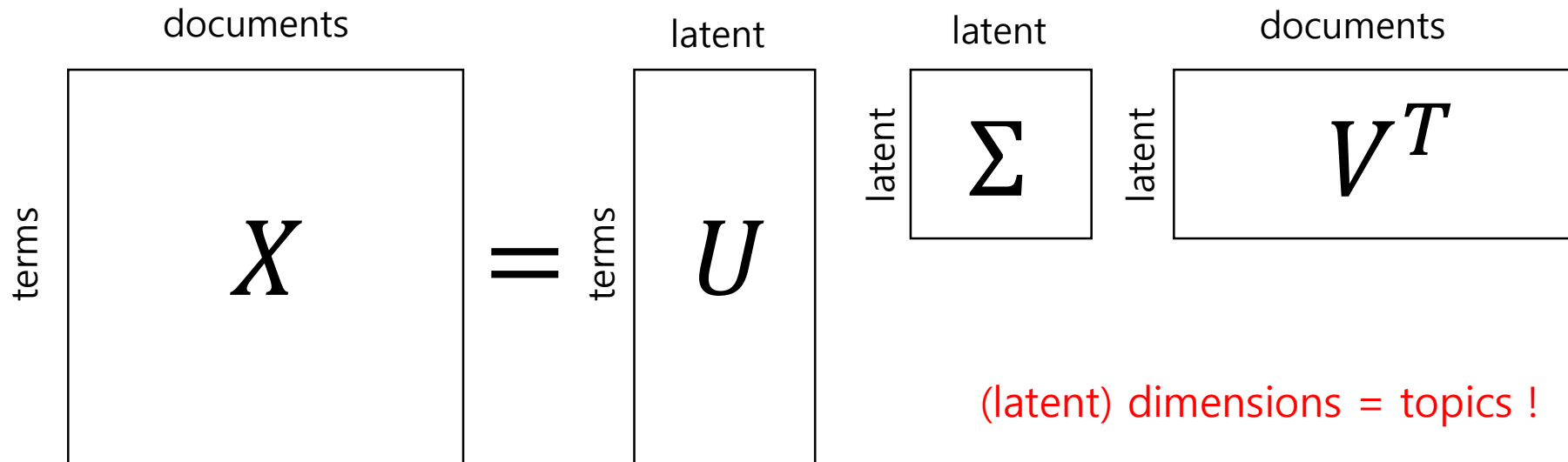
- LSI 는 함께 등장하는 단어 집합을 하나의 개념 (토픽)으로 묶는 효과가 있습니다. (co-occurrence)



# Latent Semantic Indexing (LSI)

---

- LSI 는  $k$  차원의 벡터로 단어와 문서를 표현합니다.
- Topic 에 관련된 정보를 보존하는 차원압축 방법입니다.



(latent) dimensions = topics !

# Latent Semantic Indexing (LSI)

First two topics much more important than other topics.

3	4	1	0
4	3	0	1
3	4	4	3
0	1	4	3
2	0	3	3
0	1	3	4

**=**

	To1	To2	To3	To4
Te1	-0.33	-0.53	0.37	-0.14
Te2	-0.32	-0.54	-0.49	0.35
Te3	-0.62	-0.10	0.26	-0.14
Te4	-0.38	0.42	0.30	-0.24
Te5	-0.36	0.25	-0.68	-0.47
Te6	-0.37	0.42	0.02	0.75

**X**

Topic Importance

11.4			
	6.27		
		2.22	
			1.28

**X**

	D1	D2	D3	D4
To1	-0.42	-0.48	-0.57	-0.51
To2	-0.56	-0.52	0.45	0.46
To3	-0.65	0.62	0.28	-0.35
To4	-0.30	0.34	-0.63	0.63

# Latent Semantic Indexing (LSI)

두 components 에 Term 1, 2 의 유사성이 보존됩니다.

3	4	1	0
4	3	0	1
3	4	4	3
0	1	4	3
2	0	3	3
0	1	3	4

=

	To1	To2	To3	To4
Te1	-0.33	-0.53	0.37	-0.14
Te2	-0.32	-0.54	-0.49	0.35
Te3	-0.62	-0.10	0.26	-0.14
Te4	-0.38	0.42	0.30	-0.24
Te5	-0.36	0.25	-0.68	-0.47
Te6	-0.37	0.42	0.02	0.75

X

Topic Importance

11.4			
	6.27		
		2.22	
			1.28

X

	D1	D2	D3	D4
To1	-0.42	-0.48	-0.57	-0.51
To2	-0.56	-0.52	0.45	0.46
To3	-0.65	0.62	0.28	-0.35
To4	-0.30	0.34	-0.63	0.63

# Latent Semantic Indexing (LSI)

두 components 에 Term 4, 5, 6 의 유사성이 보존됩니다.

3	4	1	0
4	3	0	1
3	4	4	3
0	1	4	3
2	0	3	3
0	1	3	4

=

	To1	To2	To3	To4
Te1	-0.33	-0.53	0.37	-0.14
Te2	-0.32	-0.54	-0.49	0.35
Te3	-0.62	-0.10	0.26	-0.14
Te4	-0.38	0.42	0.30	-0.24
Te5	-0.36	0.25	-0.68	-0.47
Te6	-0.37	0.42	0.02	0.75

X

Topic Importance

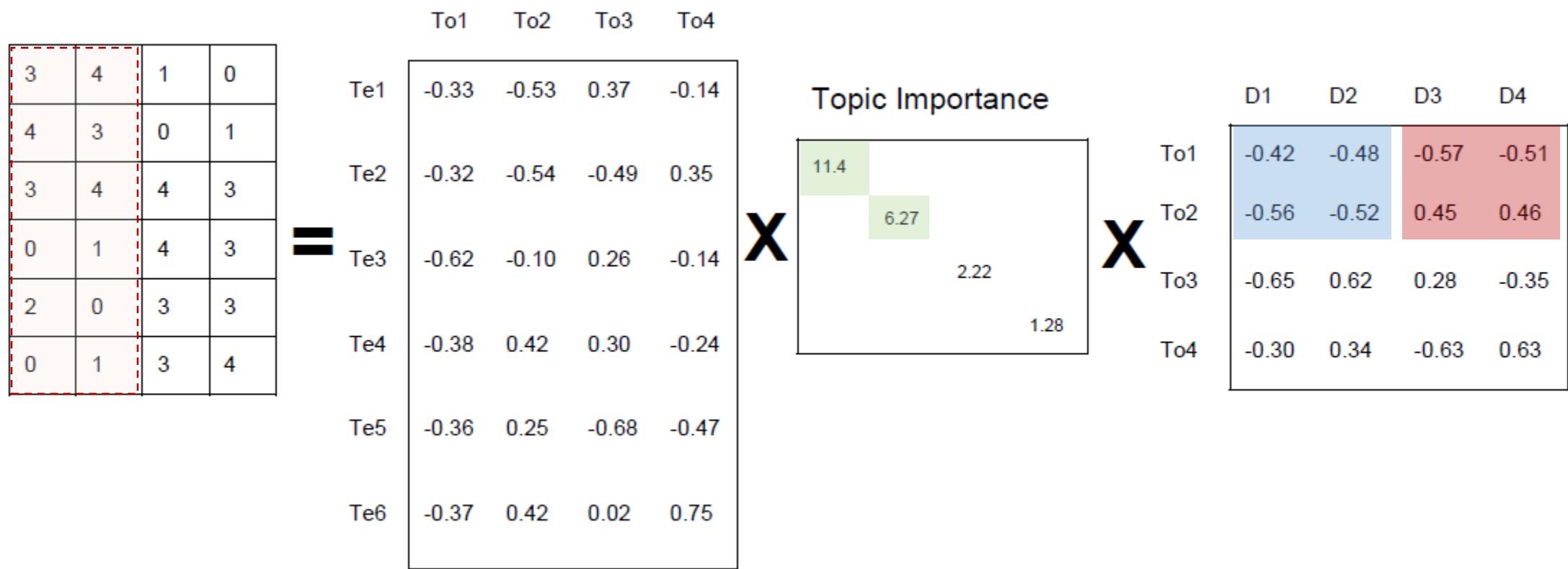
11.4			
	6.27		
		2.22	
			1.28

X

	D1	D2	D3	D4
To1	-0.42	-0.48	-0.57	-0.51
To2	-0.56	-0.52	0.45	0.46
To3	-0.65	0.62	0.28	-0.35
To4	-0.30	0.34	-0.63	0.63

# Latent Semantic Indexing (LSI)

두 components 에 doc 1, 2 / doc 3, 4 의 유사성이 보존됩니다.



# Latent Semantic Indexing (LSI)

---

- Co-occurrence 가 높은 단어들을 비슷한 topical vector 로 표현됩니다.
- 하지만 Word2Vec 처럼 음의 값이 있기 때문에 해석이 어렵습니다.

	To1	To2	To3	To4
Te1	-0.33	-0.53	0.37	-0.14
Te2	-0.32	-0.54	-0.49	0.35
Te3	-0.62	-0.10	0.26	-0.14
Te4	-0.38	0.42	0.30	-0.24
Te5	-0.36	0.25	-0.68	-0.47
Te6	-0.37	0.42	0.02	0.75

# Probabilistic Latent Semantic Indexing

---

- pLSI 혹은 pLSA 라 불립니다.
- LSI 는 단어 – 문서 행렬로부터, 단어와 문서에 대한  $k$  차원의 topical representation을 학습합니다.
  - Word2Vec 처럼 각 벡터의 값으로부터 특정한 의미를 해석하기 어렵습니다.



# Probabilistic Latent Semantic Indexing

---

- pLSI 는 확률 모델을 이용하여 단어와 문서를 표현합니다.
  - 각 문서는 topic probability vector 로 표현되며,
  - Topic vector 도 bag of words 와 비슷한 단어 벡터 형태로 표현됩니다.
  - 모델의 해석에 유용합니다.

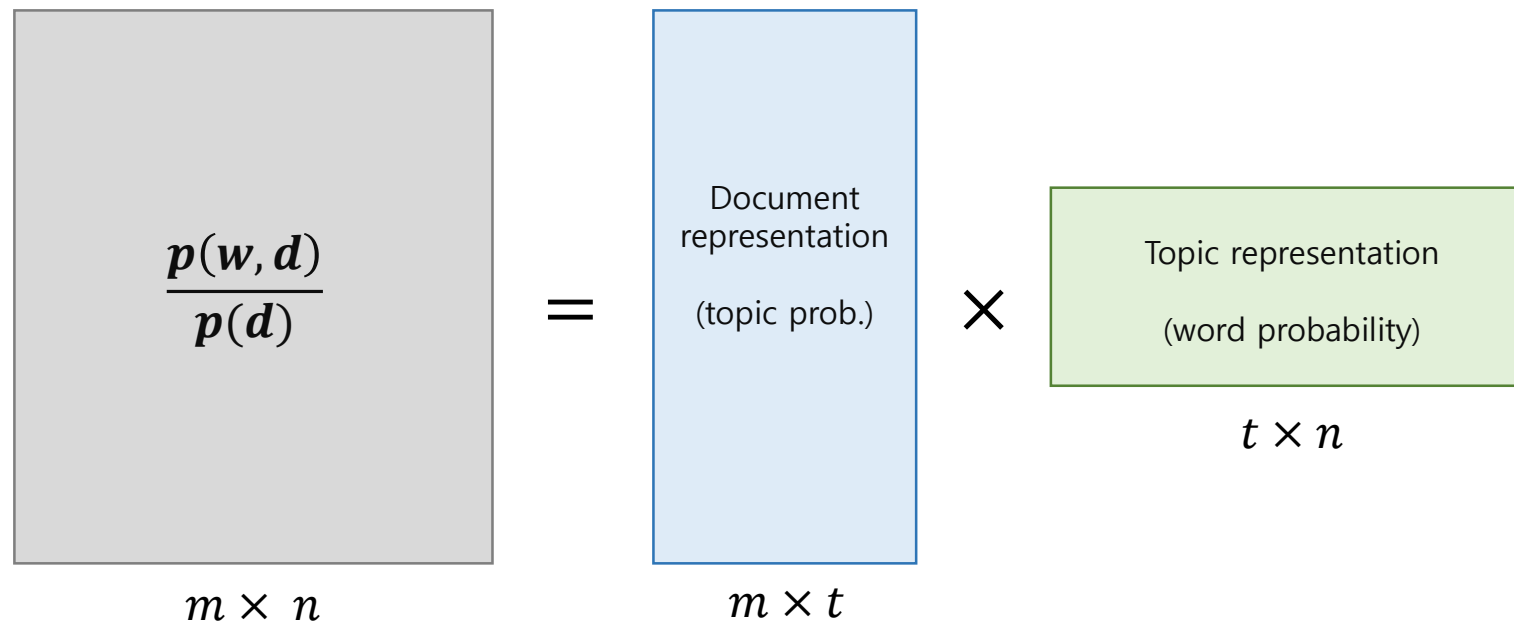
# Probabilistic Latent Semantic Indexing

---

- pLSI 는 문서  $d$  에서 단어  $w$  가 등장할 확률을 학습합니다.
- 문서  $d$  에서 단어  $w$  가 만들어지기 위해서 토픽  $z$  이라는 숨겨진 변수가 있다고 가정합니다.
  - $p(w, d) = p(d) \times \frac{p(w, d)}{p(d)} = p(d) \times p(w|d)$
  - $p(w, d) = p(d) \sum_z p(w|z) \times p(z|d)$

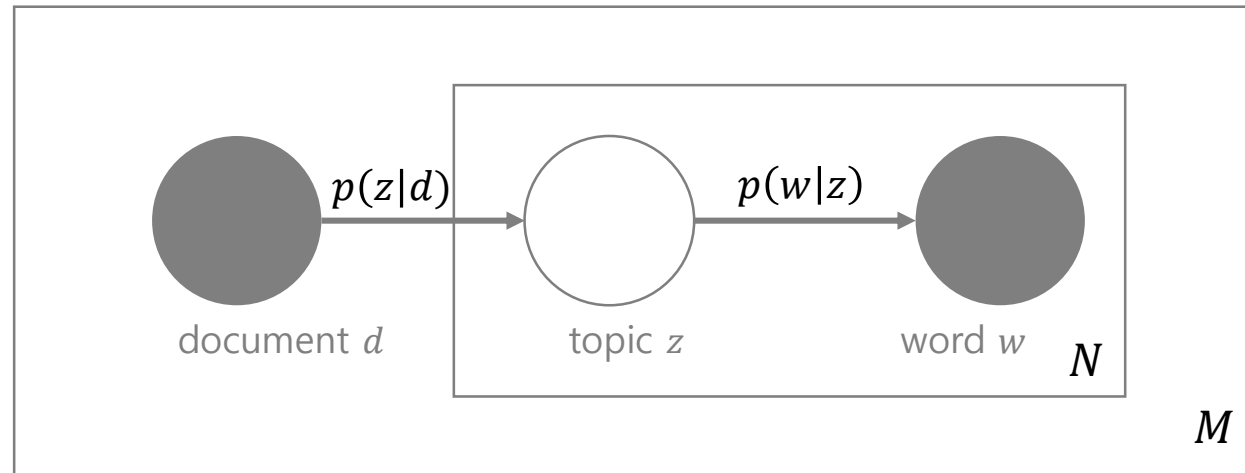
# Probabilistic Latent Semantic Indexing

- $p(w, d) = p(d) \sum_z p(w|z) \times p(z|d)$ 
  - $p(z|d)$ : 문서  $d$  에서 토픽  $z$  가 발생할 확률
  - $p(w|z)$ : 토픽  $z$  에서 단어  $w$  가 발생할 확률
  - $p(z|d), p(w|z)$  는 multinomial distribution 을 이용합니다.



# Probabilistic Latent Semantic Indexing

- pLSI 는 graphical model representation 으로 표현됩니다.
- $p(w, d) = p(d) \sum_z p(w|z) \times p(z|d)$



○ hidden variables

● observed variables

$N$  : number of words

$M$  : number of docs

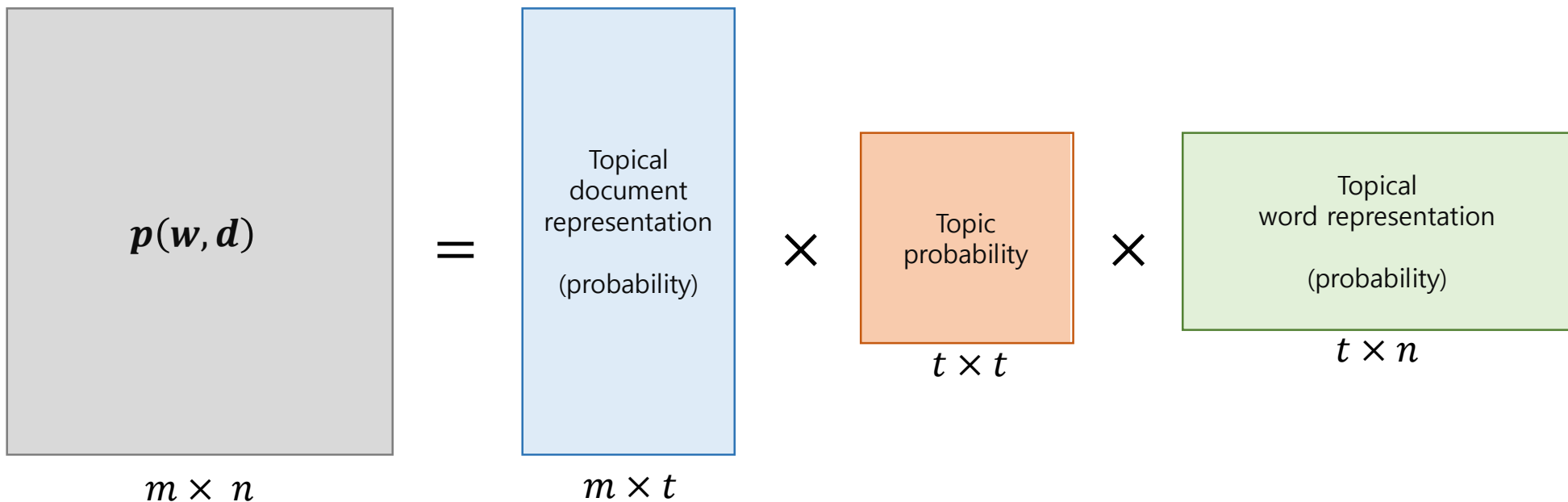
# Probabilistic Latent Semantic Indexing

---

- pLSI 를 표현하는 방법은 두 가지 입니다
  - asymmetric :  $p(w, d) = p(d) \sum_z p(w|z) \times p(z|d)$
  - symmetric :  $p(w, d) = \sum_z p(w|z) \times p(d|z) \times p(z)$
- 두 가지 방법은 서로의 목적에 따라 적절한 모델로 선택하면 됩니다

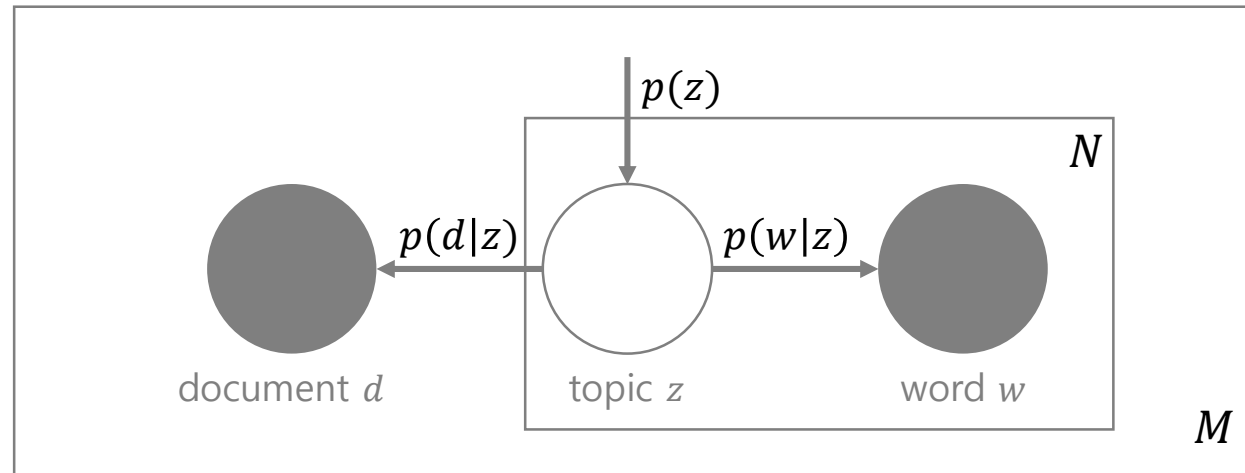
# Probabilistic Latent Semantic Indexing

- $p(w, d) = \sum_z p(w|z) \times p(z|d) \times p(z)$ 
  - $p(d|z)$ : 토픽  $z$  에서 문서  $d$  가 발생할 확률
  - $p(w|z)$ : 토픽  $z$  에서 단어  $w$  가 발생할 확률
  - $p(z)$  : 토픽  $z$  의 확률



# Probabilistic Latent Semantic Indexing

- Symmetric pLSI 도 graphical model representation 으로 표현됩니다.
  - $p(w, d) = \sum_z p(w|z) \times p(d|z) \times p(z)$



○ hidden variables

● observed variables

$N$  : number of words

$M$  : number of docs

# Probabilistic Latent Semantic Indexing

---

- $p(w, d) = p(d) \sum_z p(w|z) \times p(z|d)$  or  
 $p(w, d) = \sum_z p(w|z) \times p(d|z) \times p(z)$
- $z$  는 학습데이터에 드러나지 않은 latent variables 입니다.
  - 학습이 어렵습니다.
  - Expected – Maximization 이라는 방법이 이용됩니다.



# Probabilistic Latent Semantic Indexing

---

- pLSI 는 새로운 문서에 대한  $p(z|d)$  를 학습하기 어렵습니다.
- pLSI 는 학습할 패러미터가 많아 over-fitting 이 일어날 가능성이 높습니다.
- Latent Dirichlet Allocation (LDA) 는 위 문제들을 해결하기 위하여 제안된 방법입니다.

# Latent Dirichlet Allocation

---

- LDA 는 topic modeling 을 위하여 가장 많이 이용되는 알고리즘입니다.
- Multinomial 을 이용하는 pLSI 와 달리 사용자 정의 parameter ( $\alpha, \beta$ ) 를 따르는 Dirichlet distribution 을 이용합니다.

- pLSI :  $p(w, d) = p(d) \sum_z p(z|d) \times p(w|z)$

- LDA :  $p(\boldsymbol{\theta}, \boldsymbol{\phi}, \mathbf{z}, \mathbf{w} | \alpha, \beta) = p(\boldsymbol{\theta}|\alpha) \times p(\boldsymbol{\phi}|\beta) \times \prod_{n=1}^N p(z_n|\boldsymbol{\theta})p(w_n|\boldsymbol{\phi}, z_n)$

# Latent Dirichlet Allocation

---

- LDA 의 학습 결과로 두 가지 정보를 얻을 수 있습니다.

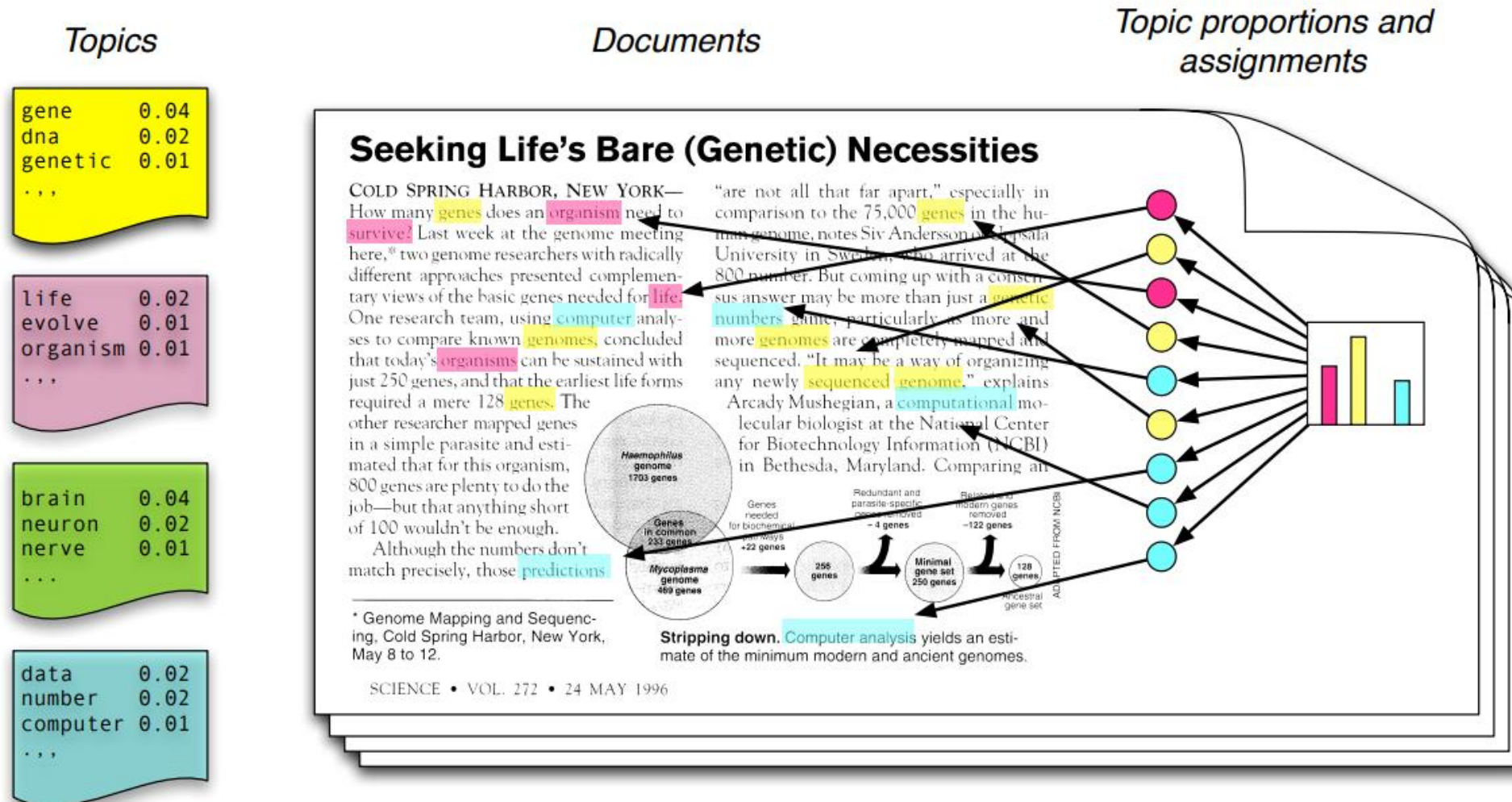
(1) 각 문서의 토픽 확률 분포

(2) 각 토픽의 단어 확률 분포

$$p(\boldsymbol{\theta}, \boldsymbol{\phi}, \mathbf{z}, \mathbf{w} \mid \alpha, \beta) = p(\boldsymbol{\theta} \mid \alpha) \times p(\boldsymbol{\phi} \mid \beta) \times \prod_{n=1}^N p(z_n \mid \boldsymbol{\theta}) p(w_n \mid \boldsymbol{\phi}, z_n)$$

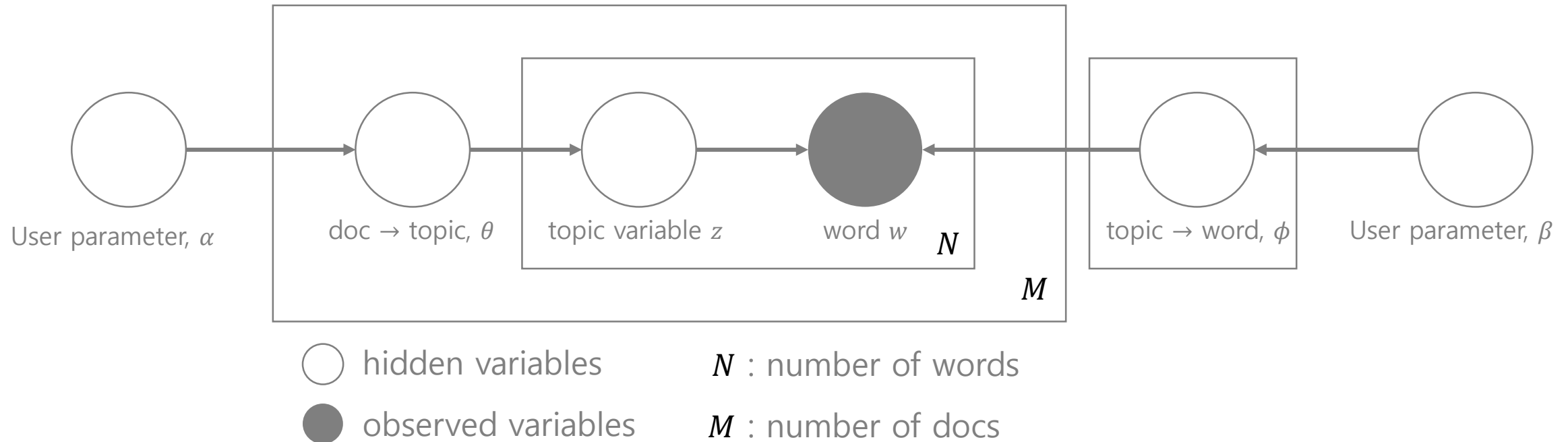
- 한 문서에서 등장한 단어들을 같은 토픽으로 묶는 효과가 있습니다.

# Latent Dirichlet Allocation



# Latent Dirichlet Allocation

- $p(\boldsymbol{\theta}, \boldsymbol{\phi}, \mathbf{z}, \mathbf{w} | \alpha, \beta) = p(\boldsymbol{\theta} | \alpha) \times p(\boldsymbol{\phi} | \beta) \times \prod_{n=1}^N p(z_n | \boldsymbol{\theta}) p(w_n | \boldsymbol{\phi}, z_n)$



# Latent Dirichlet Allocation

---

- LDA 의 학습에 대한 설명은 “밑바닥부터 시작하는 데이터 과학” 책을 참고하시면 좋습니다.
  - 해당 책의 LDA 구현 코드를 통하여 설명합니다.
  - 이 내용은 아래 블로그에도 잘 정리되어 있습니다.

# Latent Dirichlet Allocation

---

- 그 외의 LDA 의 학습 과정에 관련한 직관적인 설명들은 아래의 블로그를 참고하세요.

# Latent Dirichlet Allocation

- LDA 는 junk topic & term 이 존재합니다.
- IMDB 영화리뷰를 이용한 LDA 학습 후, 각 토픽 별 발생확률이 가장 높은 단어들은 정보력이 적고 흔한 단어입니다.

Topic #1	Topic #2	Topic #3	Topic #4	Topic #5
the (0.0395)	the (0.0773)	the (0.0441)	the (0.0489)	the (0.0462)
and (0.0280)	and (0.0313)	to (0.0274)	of (0.0286)	of (0.0316)
to (0.0265)	to (0.0255)	of (0.0234)	to (0.0282)	to (0.0261)
of (0.0255)	of (0.0222)	is (0.0230)	and (0.0215)	is (0.0241)
is (0.0180)	in (0.0212)	and (0.0226)	it (0.0182)	and (0.0181)
in (0.0156)	is (0.0162)	in (0.0152)	is (0.0172)	in (0.0173)
this (0.0153)	that (0.0154)	that (0.0146)	in (0.0145)	it (0.0162)
it (0.0121)	it (0.0151)	this (0.0141)	movie (0.0139)	this (0.0137)
with (0.0110)	this (0.0141)	it (0.0139)	that (0.0126)	that (0.0127)
that (0.0107)	movie (0.0101)	movie (0.0099)	this (0.0116)	movie (0.0117)



# Latent Dirichlet Allocation

---

- LDA 는 junk topic & term 이 존재합니다.
  - Junk topic 이란 많은 문서들에서 거의 이용되지 않는 topic 이나 혹은 모든 문서에서 이용되는 topic 입니다.
  - 이를 방지하기 위해서는 전처리 과정에서 불필요한 단어들을 미리 제거하는 것이 좋습니다.
    - Term / document frequency filtering
    - Stopwords removal

# LDA vs Word2Vec

---

- LDA 와 Word2Vec 은 모두 단어를 벡터 공간의 좌표로 표현합니다.
  - Word2Vec 의 공간은 각 차원을 해석하기 어렵습니다만, LDA 는 확률을 표현하는 벡터이기 때문에 해석이 가능합니다.
- Word2Vec 은 좌/우 단어의 분포를 contexts 로 이용합니다만, LDA 는 한 문서에서 함께 등장한 co-occurrences 를 contexts 로 이용합니다.
  - Topically similar vs. contextually similar

# Latent Dirichlet Allocation

---

- Topic 의 개수는 일단 크게 잡는 쪽이 좋습니다.
  - Perplexity 와 같은 evaluation metric 이 있습니다만, 현실적인 솔루션은 일단 여러 개의 topics 을 추출하고, 비슷한 topics 은 후처리로 묶는 것이 좋습니다.
  - topic 개수가 작으면 여러 개의 topic 이 하나로 뭉치는 경향도 있습니다.
  - LDA 는 co-occurrence 를 이용하는 단어의 군집화와 비슷합니다.

# pyLDavis

---

- LDA 의 학습 결과를 시각적으로 표현합니다.
  - LDA 모델은 토픽의 단어 확률을 학습합니다.
  - $t \times n$  크기의 (토픽, 단어) 벡터를 PCA 를 이용하여  $t \times 2$  로 압축합니다.
  - 비슷한 토픽은 비슷한 2 차원 좌표를 지닙니다.

# pyLDavis

---

- 토픽 별 키워드를 interactive 한 방법으로 선택합니다.
- Saliency (coverage) / distinctiveness (discriminative power) 사이의 가중치를 사용자가 조절함으로써 토픽을 잘 이해할 수 있도록 돕습니다.
  - Saliency :  $P(word | topic)$
  - Distinctiveness :  $\frac{P(word | topic)}{P(word)}$
  - Keyword score :  $\lambda \times p(word | topic) + (1 - \lambda) \times \frac{P(word | topic)}{P(word)}$

# pyLDAvis

In [15]: pyLDAvis.display(loaded\_pyldavis)

Out [15]:

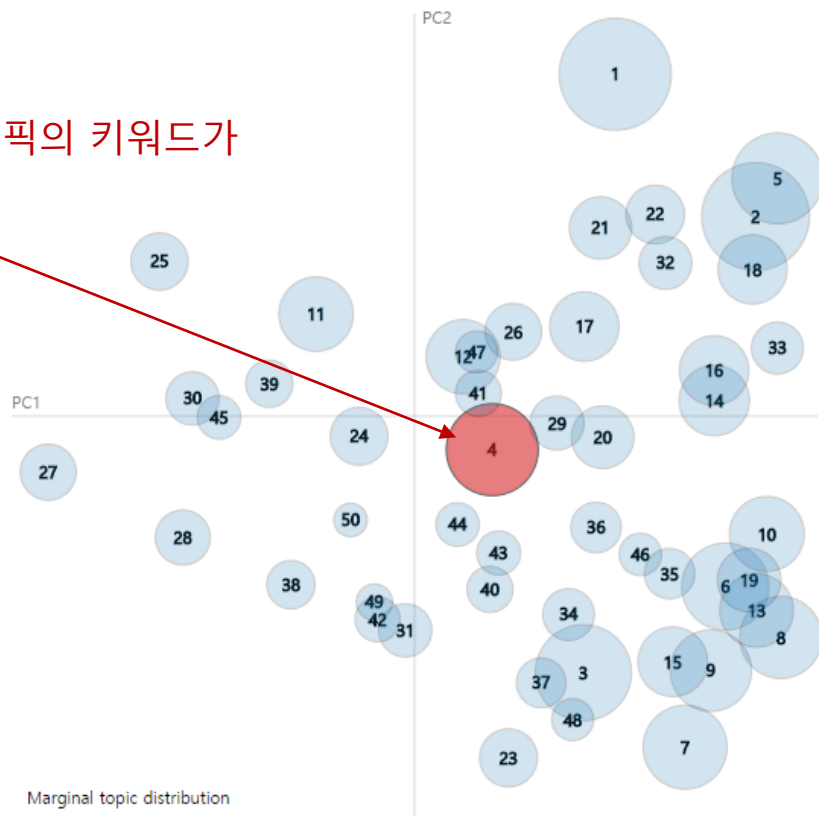
Selected Topic: 4 Previous Topic Next Topic Clear Topic

Slide to adjust relevance metric:(2)

$\lambda = 1$

0.0 0.2 0.4 0.6 0.8 1

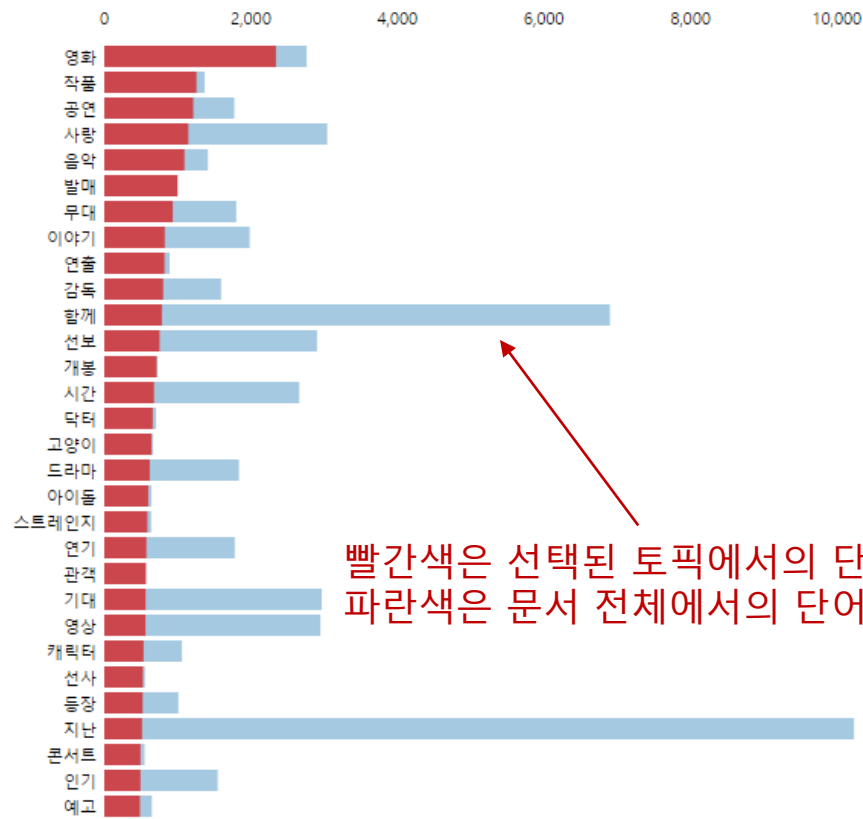
Intertopic Distance Map (via multidimensional scaling)



Marginal topic distribution



Top-30 Most Relevant Terms for Topic 4 (4.1% of tokens)



빨간색은 선택된 토픽에서의 단어 등장 비율,  
파란색은 문서 전체에서의 단어 등장 비율입니다

Overall term frequency  
Estimated term frequency within the selected topic

1.  $\text{saliency}(\text{term } w) = \text{frequency}(w) * [\sum_t p(t | w) * \log(p(t | w)/p(t))]$  for topics  $t$ ; see Chuang et al (2012)  
2.  $\text{relevance}(\text{term } w | \text{topic } t) = \lambda * p(w | t) + (1 - \lambda) * p(w | t)/p(w)$ ; see Sievert & Shirley (2014)

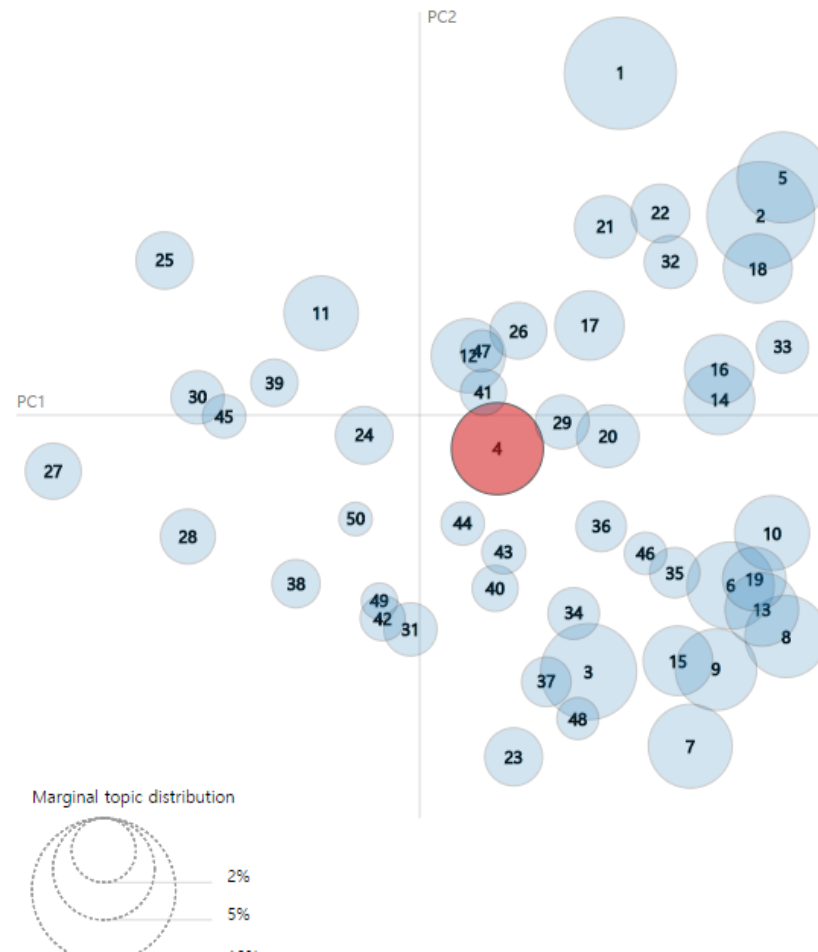
# pyLDavis

```
In [15]: pyLDavis.display(loaded_pyldavis)
```

```
Out [15]:
```

Selected Topic:  Previous Topic Next Topic Clear Topic

Intertopic Distance Map (via multidimensional scaling)

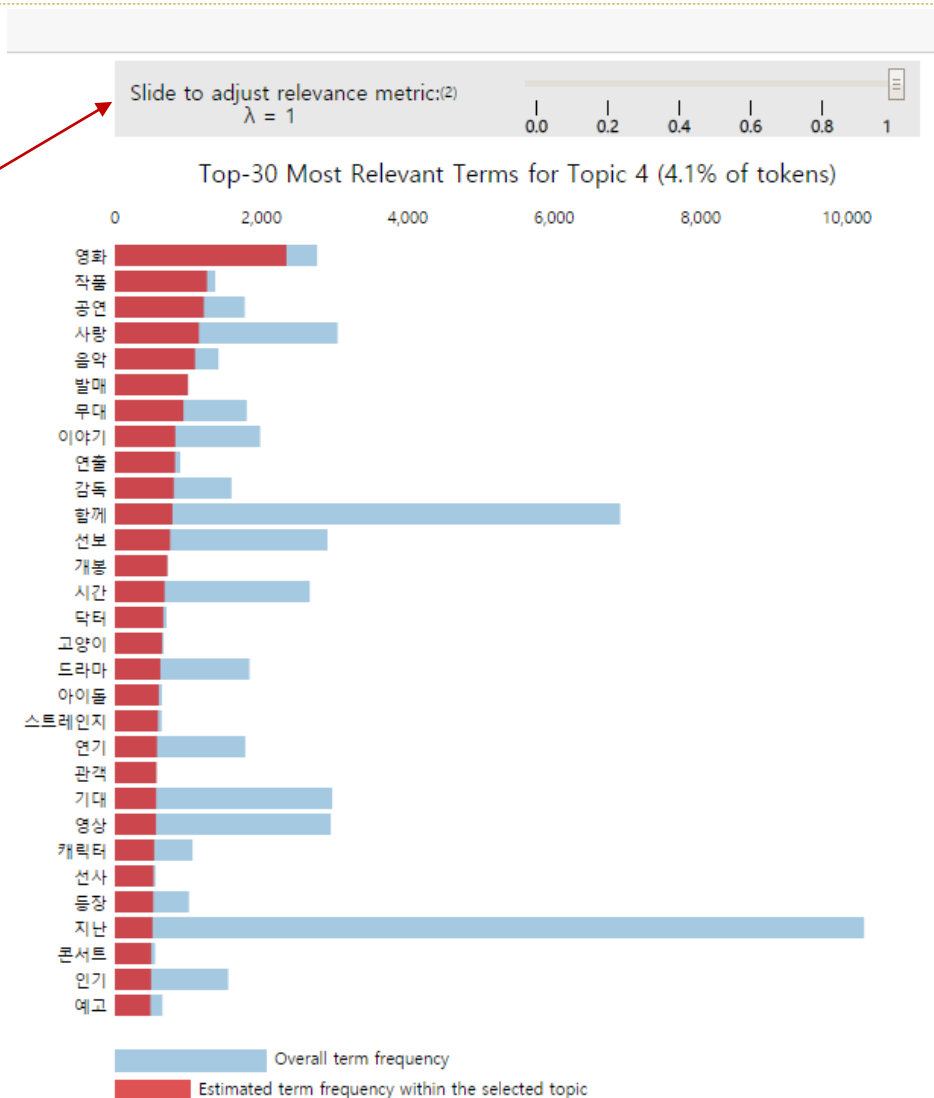


- PCA 를 통하여 (topic, word) 행렬이 (topic, 2) 로 압축됩니다.

# pyLDAvis

- score :  $\lambda \times p(\text{word} \mid \text{topic}) + (1 - \lambda) \times \frac{P(\text{word} \mid \text{topic})}{P(\text{word})}$

$\lambda$  에 의하여 saliency 와 distinctiveness 의  
중요도가 조절됩니다



1. saliency(term w) = frequency(w) \* [sum\_t p(t | w) \* log(p(t | w)/p(t))]; for topics t; see Chuang et. al (2012)  
2. relevance(term w | topic t) =  $\lambda * p(w | t) + (1 - \lambda) * p(w | t)/p(w)$ ; see Sievert & Shirley (2014)



# History

---

Latent Semantic Indexing  
(LSI), 1990

- Grouping semantically similar terms to a topic
- Using matrix decomposition

Probabilistic LSI  
(pLSI), 1999

- Using probabilistic model

Latent Dirichlet Allocation  
(LDA), 2003

- Using (more general) probabilistic model

now

- Using distributed representation