6. 토픽모델링

Topic Modeling

집현전 초급반 6조

김준태, 박준현, 전인성

목차

잠재 의미 분석

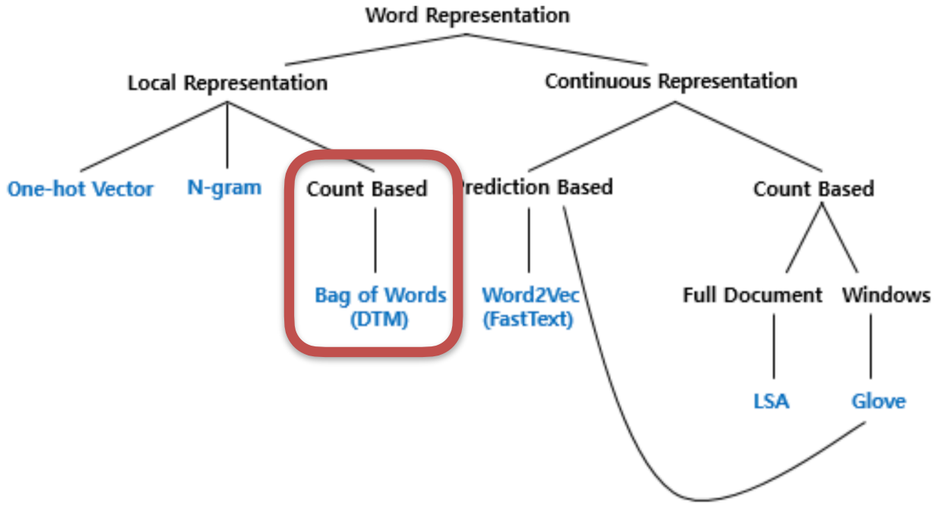
Latent Semantic Analysis

Motivation

▶ 문서를 표현하는 가장 직관적이고 쉬운 방법은 카운트 기반의 Bag of Words

▶ Document-Term Matrix, DTM

-	과일이	길고	노란	먹고	바나나	사과	싶은	저는	좋아요
문서1	0	0	0	1	0	1	1	0	0
문서2	0	0	0	1	1	0	1	0	0
문서3	0	1	1	0	2	0	0	0	0
문서4	1	0	0	0	0	0	0	1	1



▶ Term-Frequency-Inverse-Term-Frequency, TF-IDF

-	과일이	길고	노란	먹고	바나나	사과	싶은	저는	좋아요
문서1	0	0	0	0.287682	0	0.693147	0.287682	0	0
문서2	0	0	0	0.287682	0.287682	0	0.287682	0	0
문서3	0	0.693147	0.693147	0	0.575364	0	0	0	0
문서4	0.693147	0	0	0	0	0	0	0.693147	0.693147

Motivation

▶ 그러나 DTM이나 TF-IDF는 단어의 의미(관계, 유사도, 토픽)를 고려하지 못함.

▶ Document-Term Matrix, DTM

-	과일이	길고	노란	먹고	바나나	사과	싶은	저는	좋아요
문서1	0	0	0	1	0	1	1	0	0
문서2	0	0	0	1	1	0	1	0	0
문서3	0	1	1	0	2	0	0	0	0
문서4	1	0	0	0	0	0	0	1	1

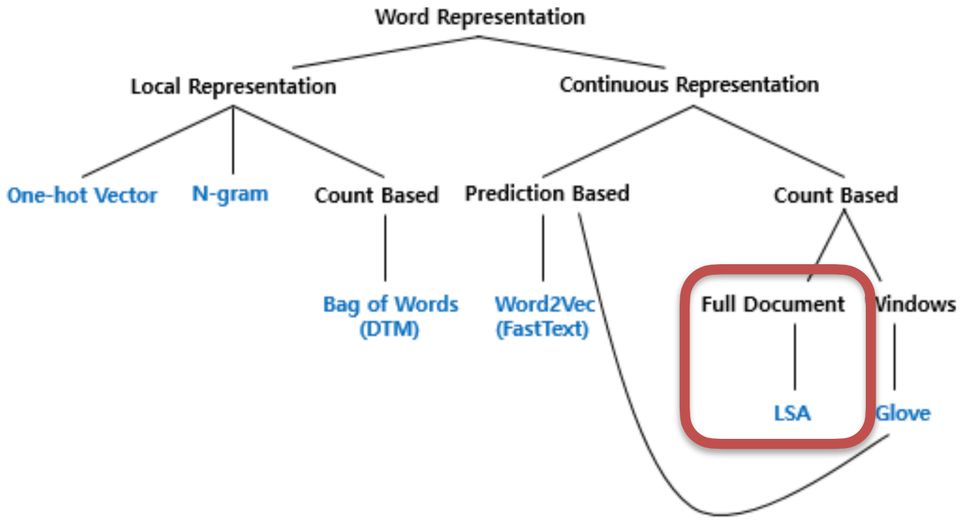
토픽 : 과일

Motivation

▶ 그러나 DTM이나 TF-IDF는 단어의 의미(관계, 유사도, 토픽)를 고려하지 못함.

▶ Document-Term Matrix, DTM

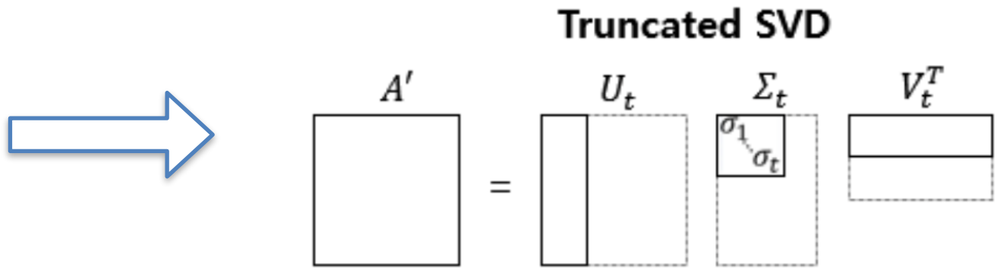
-	과일이	길고	노란	먹고	바나나	사과	싶은	저는	좋아요
문서1	0	0	0	1	0	1	1	0	0
문서2	0	0	0	1	1	0	1	0	0
문서3	0	1	1	0	2	0	0	0	0
문서4	1	0	0	0	0	0	0	1	1



잠재 의미 분석(Latent Semantic Analysis, LSA)

- DTM, TF-IDF에서 시작하는 카운트 기반 의미 분석
- 문서 단어 행렬(DTM)이나 단어 빈도-역 문서 빈도(TF-IDF) 행렬을 특이값 분해(Singular Value Decomposition, SVD)를 통해 Low-rank approximation 수행

-	과일이	길고	노란	먹고	바나나	사과	싶은	저는	좋아요
문서1	0	0	0	1	0	1	1	0	0
문서2	0	0	0	1	1	0	1	0	0
문서3	0	1	1	0	2	0	0	0	0
문서4	1	0	0	0	0	0	0	1	1



- 문서 단어 행렬의 크기 축소
- 중요한 정보만 남기기
- 비슷한 단어들을 토픽으로 묶어서 문서 분석

특이값 분해(Singular Value Decomposition, SVD)

▶ A 라는 $m \times n$ 행렬이 있을 때 다음과 같이 3개의 행렬의 곱으로 분해함

▶ $A = U\Sigma \mathrm{V}^T = (\mathrm{m} \times m) \times (m \times n) \times (n \times n)^T$

✓$U = m \times m$ 직교행렬

- ✓ 왼쪽 특이행렬
- ✓ $UU^T = I,\ U^{-1} = U^T$

$$\begin{bmatrix} 2 & 3 \\ 1 & 4 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0.82 & -0.58 & 0 & 0 \\ 0.58 & 0.82 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 5.47 & 0 & 0 & 0 \\ 0 & 0.37 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0.40 & 0.91 \\ -0.91 & 0.40 \end{bmatrix}$$

✓$\Sigma = m \times n$ 직사각 대각행렬

- ✓ 대각선 말고는 다 0.
- ✓ 대각값들은 A의 특이값(singular value)이라고 함.

✓$V = n \times n$ 직교행렬

- ✓ 오른쪽 특이행렬
- ✓ $V^T = V$의 전치행렬

특이값 분해(Singular Value Decomposition, SVD)

▶ A 라는 $m \times n$ 행렬이 있을 때 다음과 같이 3개의 행렬의 곱으로 분해함

▶ $A = U\Sigma \mathrm{V}^T = (\mathrm{m} \times m) \times (m \times n) \times (n \times n)^T$

$$\begin{bmatrix} 2 & 3 \\ 1 & 4 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0.82 & -0.58 & 0 & 0 \\ 0.58 & 0.82 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 5.47 & 0 & 0 & 0 \\ 0 & 0.37 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0.40 & 0.91 \\ -0.91 & 0.40 \end{bmatrix}$$

A U Σ V^T

특이값 분해(Singular Value Decomposition, SVD)

- A 라는 $m \times n$ 행렬이 있을 때 다음과 같이 3개의 행렬의 곱으로 분해함

- $A = U\Sigma \mathrm{V}^T = (\mathrm{m} \times m) \times (m \times n) \times (n \times n)^T$

$$\begin{bmatrix} 2 & 3 \\ 1 & 4 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0.82 & -0.58 & 0 & 0 \\ 0.58 & 0.82 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 5.47 & 0 & 0 & 0 \\ 0 & 0.37 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0.40 & 0.91 \\ -0.91 & 0.40 \\ & \\ & \end{bmatrix}$$

A U Σ V^T

특이값 분해(Singular Value Decomposition, SVD)

▶ A 라는 $m \times n$ 행렬이 있을 때 다음과 같이 3개의 행렬의 곱으로 분해함

▶ $A = U\Sigma V^T = (\mathrm{m} \times m) \times (m \times n) \times (n \times n)^T$

$$\begin{bmatrix} 2 & 3 \\ 1 & 4 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0.82 & -0.58 & 0 & 0 \\ 0.58 & 0.82 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 5.47 & 0 & 0 & 0 \\ 0 & 0.37 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0.40 & 0.91 \\ -0.91 & 0.40 \end{bmatrix}$$

(u_1: 첫 번째 열, σ_1: 첫 번째 대각 원소, v_1: 첫 번째 행)

$$A = u_1 \sigma_1 v_1 + u_2 \sigma_2 v_2 + \dots$$

특이값 분해(Singular Value Decomposition, SVD)

- A 라는 $m \times n$ 행렬이 있을 때 다음과 같이 3개의 행렬의 곱으로 분해함

- $A = U\Sigma \mathrm{V}^T = (\mathrm{m} \times m) \times (m \times n) \times (n \times n)^T$

$$\begin{bmatrix} 2 & 3 \\ 1 & 4 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0.82 & -0.58 & 0 & 0 \\ 0.58 & 0.82 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 5.47 & 0 & 0 & 0 \\ 0 & 0.37 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0.40 & 0.91 \\ -0.91 & 0.40 \end{bmatrix}$$

A = content 1 + content 2 + …

특이값 분해 (SVD)

- A 라는 $m \times n$ 행렬이 있을 때 다음과 같이 3개의 행렬의 곱으로 분해함

- $A = U\Sigma \mathrm{V}^T = (\mathrm{m} \times m) \times (m \times n) \times (n \times n)^T$

$$\begin{bmatrix} 2 & 3 \\ 1 & 4 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0.82 & -0.58 & 0 & 0 \\ 0.58 & 0.82 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 5.47 & 0 & 0 & 0 \\ 0 & 0.37 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0.40 & 0.91 \\ -0.91 & 0.40 \end{bmatrix}$$

$$A = u_1 \sigma_1 v_1 + u_2 \sigma_2 v_2 + \dots$$

content 1

content 2

특이값 분해 (SVD)

- A 라는 $m \times n$ 행렬이 있을 때 다음과 같이 3개의 행렬의 곱으로 분해함

- $A = U\Sigma \mathrm{V}^T = (\mathrm{m} \times m) \times (m \times n) \times (n \times n)^T$

$$\begin{bmatrix} 2 & 3 \\ 1 & 4 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0.82 & -0.58 & 0 & 0 \\ 0.58 & 0.82 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 5.47 & 0 & 0 & 0 \\ 0 & 0.37 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0.40 & 0.91 \\ -0.91 & 0.40 \end{bmatrix}$$

$$A = u_1 \sigma_1 v_1 + u_2 \sigma_2 v_2$$

content 1

content 2

절단된 특이값 분해 (truncated SVD)

▶ truncated SVD : SVD를 수행한 다음, 중요한 컨텐츠만 남겨서 정보를 압축하기

$$\begin{bmatrix} 2 & 3 \\ 1 & 4 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0.82 & -0.58 & 0 & 0 \\ 0.58 & 0.82 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 5.47 & 0 & 0 & 0 \\ 0 & 0.37 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0.40 & 0.91 \\ -0.91 & 0.40 \end{bmatrix}$$

$$A = u_1 \sigma_1 v_1 + u_2 \sigma_2 v_2$$

content 1

content 2

절단된 특이값 분해 (truncated SVD)

▶ truncated SVD : SVD를 수행한 다음, 중요한 컨텐츠만 남겨서 정보를 압축하기

▶ $A \approx \hat{U}\hat{\Sigma}\hat{V}^T = (m \times t) \times (t \times t) \times (t \times n)$: t 개의 컨텐츠만 남기는 경우

$$\begin{bmatrix} 2 & 3 \\ 1 & 4 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0.82 & -0.58 & 0 & 0 \\ 0.58 & 0.82 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 5.47 & 0 & 0 & 0 \\ 0 & 0.37 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0.40 & 0.91 \\ -0.91 & 0.40 \end{bmatrix}$$

$$A \approx u_1 \quad \sigma_1 \quad v_1$$

content 1

잠재 의미 분석, LSA

▶ DTM이나 TF-IDF 행렬에 절단된 SVD(truncated SVD)를 사용하여 차원을 축소시키고, 단어들의 잠재적인 의미를 끌어냄

-	과일이	길고	노란	먹고	바나나	사과	싶은	저는	좋아요
문서1	0	0	0	1	0	1	1	0	0
문서2	0	0	0	1	1	0	1	0	0
문서3	0	1	1	0	2	0	0	0	0
문서4	1	0	0	0	0	0	0	1	1

```
[ ]  1 A=np.array([[0,0,0,1,0,1,1,0,0],[0,0,0,1,1,0,1,0,0],[0,1,1,0,2,0,0,0,0],[1,0,0,0,0,0,0,1,1]])
```

```
[ ]  1 A
```

```
array([[0, 0, 0, 1, 0, 1, 1, 0, 0],
       [0, 0, 0, 1, 1, 0, 1, 0, 0],
       [0, 1, 1, 0, 2, 0, 0, 0, 0],
       [1, 0, 0, 0, 0, 0, 0, 1, 1]])
```

```
[ ]  1 np.shape(A)
```

```
(4, 9)
```

잠재 의미 분석, LSA

▶ $A[4,9] \rightarrow$ np.linalg.svd $\rightarrow U[4,4], s[4], VT[9,9]$

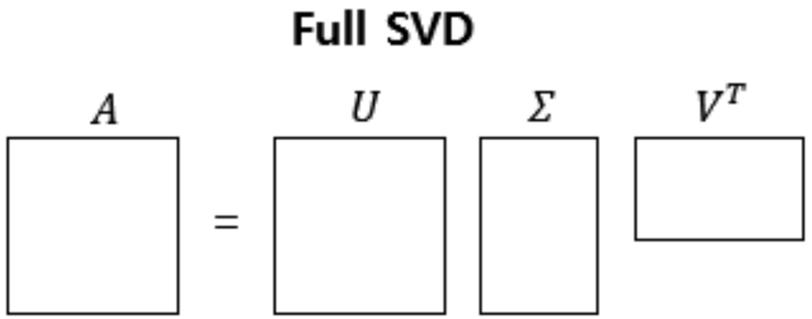
```
[4]   1 A

      array([[0, 0, 0, 1, 0, 1, 1, 0, 0],
             [0, 0, 0, 1, 1, 0, 1, 0, 0],
             [0, 1, 1, 0, 2, 0, 0, 0, 0],
             [1, 0, 0, 0, 0, 0, 0, 1, 1]])
```

```
[5]   1 U, s, VT = np.linalg.svd(A, full_matrices = True)
```

```
[9]   1 U.round(2)

      array([[-0.24,  0.75,  0.  , -0.62],
             [-0.51,  0.44, -0.  ,  0.74],
             [-0.83, -0.49, -0.  , -0.27],
             [-0.  , -0.  ,  1.  ,  0.  ]])
```

```
[10]  1 s.round(2)

      array([2.69, 2.05, 1.73, 0.77])
```

```
[11]  1 VT.round(2)

      array([[-0.  , -0.31, -0.31, -0.28, -0.8 , -0.09, -0.28, -0.  , -0.  ],
             [ 0.  , -0.24, -0.24,  0.58, -0.26,  0.37,  0.58, -0.  , -0.  ],
             [ 0.58, -0.  ,  0.  ,  0.  , -0.  ,  0.  , -0.  ,  0.58,  0.58],
             [ 0.  , -0.35, -0.35,  0.16,  0.25, -0.8 ,  0.16, -0.  , -0.  ],
             [-0.  , -0.78, -0.01, -0.2 ,  0.4 ,  0.4 , -0.2 ,  0.  ,  0.  ],
             [-0.29,  0.31, -0.78, -0.24,  0.23,  0.23,  0.01,  0.14,  0.14],
             [-0.29, -0.1 ,  0.26, -0.59, -0.08, -0.08,  0.66,  0.14,  0.14],
             [-0.5 , -0.06,  0.15,  0.24, -0.05, -0.05, -0.19,  0.75, -0.25],
             [-0.5 , -0.06,  0.15,  0.24, -0.05, -0.05, -0.19, -0.25,  0.75]])
```



잠재 의미 분석, LSA

▶ $A[4,9] \rightarrow$ np.linalg.svd $\rightarrow U[4,4], s[4], VT[9,9] \rightarrow U[4,4], S[4,4], VT[4,9]$

```
[4]  1 A

array([[0, 0, 0, 1, 0, 1, 1, 0, 0],
       [0, 0, 0, 1, 1, 0, 1, 0, 0],
       [0, 1, 1, 0, 2, 0, 0, 0, 0],
       [1, 0, 0, 0, 0, 0, 0, 1, 1]])
```

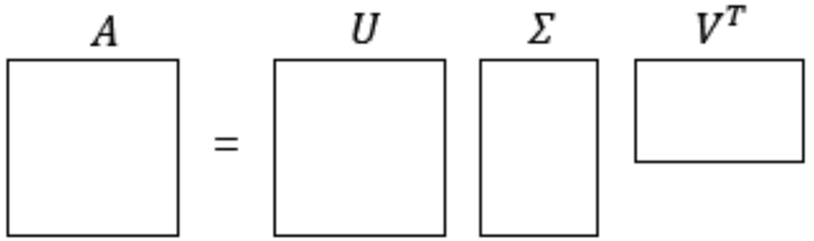
```
[5]  1 U, s, VT = np.linalg.svd(A, full_matrices = True)
```

```
[9]  1 U.round(2)

array([[-0.24,  0.75,  0.  , -0.62],
       [-0.51,  0.44, -0.  ,  0.74],
       [-0.83, -0.49, -0.  , -0.27],
       [-0.  , -0.  ,  1.  ,  0.  ]])
```

```
[10]  1 s.round(2)

array([2.69, 2.05, 1.73, 0.77])
```

```
[11]  1 VT.round(2)

array([[-0.  , -0.31, -0.31, -0.28, -0.8 , -0.09, -0.28, -0.  , -0.  ],
       [ 0.  , -0.24, -0.24,  0.58, -0.26,  0.37,  0.58, -0.  , -0.  ],
       [ 0.58, -0.  ,  0.  ,  0.  , -0.  ,  0.  , -0.  ,  0.58,  0.58],
       [ 0.  , -0.35, -0.35,  0.16,  0.25, -0.8 ,  0.16, -0.  , -0.  ],
       [-0.  , -0.78, -0.01, -0.2 ,  0.4 ,  0.4 , -0.2 ,  0.  ,  0.  ],
       [-0.29,  0.31, -0.78, -0.24,  0.23,  0.23,  0.01,  0.14,  0.14],
       [-0.29, -0.1 ,  0.26, -0.59, -0.08, -0.08,  0.66,  0.14,  0.14],
       [-0.5 , -0.06,  0.15,  0.24, -0.05, -0.05, -0.19,  0.75, -0.25],
       [-0.5 , -0.06,  0.15,  0.24, -0.05, -0.05, -0.19, -0.25,  0.75]])
```

Full SVD




```
[14]  1 S = np.diag(s)
      2
      3 S.round(2)

array([[2.69, 0.  , 0.  , 0.  ],
       [0.  , 2.05, 0.  , 0.  ],
       [0.  , 0.  , 1.73, 0.  ],
       [0.  , 0.  , 0.  , 0.77]])
```

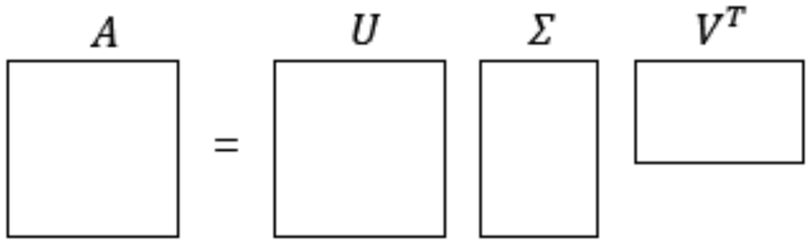
```
[15]  1 VT = VT[:4, :]
      2
      3 VT.round(2)

array([[-0.  , -0.31, -0.31, -0.28, -0.8 , -0.09, -0.28, -0.  , -0.  ],
       [ 0.  , -0.24, -0.24,  0.58, -0.26,  0.37,  0.58, -0.  , -0.  ],
       [ 0.58, -0.  ,  0.  ,  0.  , -0.  ,  0.  , -0.  ,  0.58,  0.58],
       [ 0.  , -0.35, -0.35,  0.16,  0.25, -0.8 ,  0.16, -0.  , -0.  ]])
```

잠재 의미 분석, LSA

- $A = U\Sigma V^T$

```
[18]  1 U.round(2)

      array([[-0.24,  0.75,  0.  , -0.62],
             [-0.51,  0.44, -0.  ,  0.74],
             [-0.83, -0.49, -0.  , -0.27],
             [-0.  , -0.  ,  1.  ,  0.  ]])

[19]  1 S.round(2)

      array([[2.69, 0.  , 0.  , 0.  ],
             [0.  , 2.05, 0.  , 0.  ],
             [0.  , 0.  , 1.73, 0.  ],
             [0.  , 0.  , 0.  , 0.77]])

[20]  1 VT.round(2)

      array([[-0.  , -0.31, -0.31, -0.28, -0.8 , -0.09, -0.28, -0.  , -0.  ],
             [ 0.  , -0.24, -0.24,  0.58, -0.26,  0.37,  0.58, -0.  , -0.  ],
             [ 0.58, -0.  ,  0.  ,  0.  , -0.  ,  0.  , -0.  ,  0.58,  0.58],
             [ 0.  , -0.35, -0.35,  0.16,  0.25, -0.8 ,  0.16, -0.  , -0.  ]])

[21]  1 A_recon = U @ S @ VT
      2
      3 np.abs(A_recon.round(2))

      array([[0., 0., 0., 1., 0., 1., 1., 0., 0.],
             [0., 0., 0., 1., 1., 0., 1., 0., 0.],
             [0., 1., 1., 0., 2., 0., 0., 0., 0.],
             [1., 0., 0., 0., 0., 0., 0., 1., 1.]])

[22]  1 A

      array([[0, 0, 0, 1, 0, 1, 1, 0, 0],
             [0, 0, 0, 1, 1, 0, 1, 0, 0],
             [0, 1, 1, 0, 2, 0, 0, 0, 0],
             [1, 0, 0, 0, 0, 0, 0, 1, 1]])
```

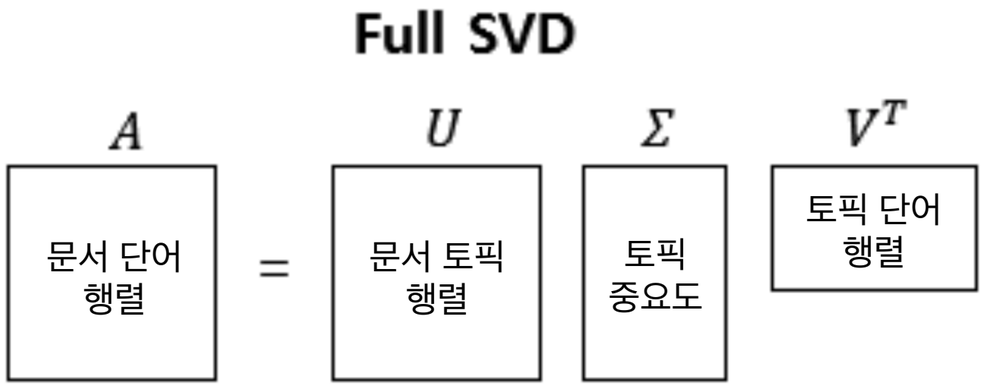
Full SVD



잠재 의미 분석, LSA

▶ Full SVD 결과

```
[18]  1 U.round(2)

      array([[-0.24,  0.75,  0.  , -0.62],
             [-0.51,  0.44, -0.  ,  0.74],
             [-0.83, -0.49, -0.  , -0.27],
             [-0.  , -0.  ,  1.  ,  0.  ]])

[19]  1 S.round(2)

      array([[2.69, 0.  , 0.  , 0.  ],
             [0.  , 2.05, 0.  , 0.  ],
             [0.  , 0.  , 1.73, 0.  ],
             [0.  , 0.  , 0.  , 0.77]])

[20]  1 VT.round(2)

      array([[-0.  , -0.31, -0.31, -0.28, -0.8 , -0.09, -0.28, -0.  , -0.  ],
             [ 0.  , -0.24, -0.24,  0.58, -0.26,  0.37,  0.58, -0.  , -0.  ],
             [ 0.58, -0.  ,  0.  ,  0.  , -0.  ,  0.  , -0.  ,  0.58,  0.58],
             [ 0.  , -0.35, -0.35,  0.16,  0.25, -0.8 ,  0.16, -0.  , -0.  ]])
```



잠재 의미 분석, LSA

▶ Σ: 토픽 중요도

```
[18]  1 U.round(2)

array([[-0.24,  0.75,  0.  , -0.62],
       [-0.51,  0.44, -0.  ,  0.74],
       [-0.83, -0.49, -0.  , -0.27],
       [-0.  , -0.  ,  1.  ,  0.  ]])
```

```
[19]  1 S.round(2)

array([[2.69, 0.  , 0.  , 0.  ],
       [0.  , 2.05, 0.  , 0.  ],
       [0.  , 0.  , 1.73, 0.  ],
       [0.  , 0.  , 0.  , 0.77]])
```

```
[20]  1 VT.round(2)

array([[-0.  , -0.31, -0.31, -0.28, -0.8 , -0.09, -0.28, -0.  , -0.  ],
       [ 0.  , -0.24, -0.24,  0.58, -0.26,  0.37,  0.58, -0.  , -0.  ],
       [ 0.58, -0.  ,  0.  ,  0.  , -0.  ,  0.  , -0.  ,  0.58,  0.58],
       [ 0.  , -0.35, -0.35,  0.16,  0.25, -0.8 ,  0.16, -0.  , -0.  ]])
```

Full SVD

A		U	Σ	V^T
문서 단어 행렬	=	문서 토픽 행렬	토픽 중요도	토픽 단어 행렬

```
S = [[토픽1, 0, 0, 0],
     [0, 토픽2, 0, 0],
     [0, 0, 토픽3, 0],
     [0, 0, 0, 토픽4]]
```

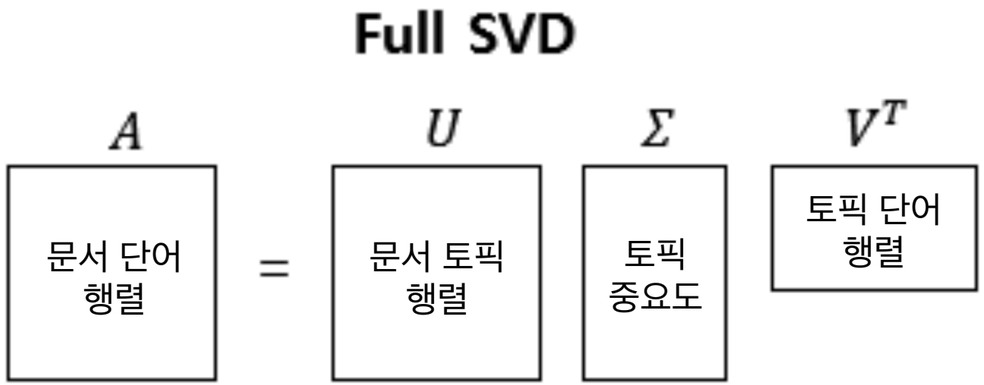
잠재 의미 분석, LSA

▶ U : 문서-토픽 행렬

```
[18]  1 U.round(2)

array([[-0.24,  0.75,  0.  , -0.62],
       [-0.51,  0.44, -0.  ,  0.74],
       [-0.83, -0.49, -0.  , -0.27],
       [-0.  , -0.  ,  1.  ,  0.  ]])
```

```
[19]  1 S.round(2)

array([[2.69, 0.  , 0.  , 0.  ],
       [0.  , 2.05, 0.  , 0.  ],
       [0.  , 0.  , 1.73, 0.  ],
       [0.  , 0.  , 0.  , 0.77]])
```

```
[20]  1 VT.round(2)

array([[-0.  , -0.31, -0.31, -0.28, -0.8 , -0.09, -0.28, -0.  , -0.  ],
       [ 0.  , -0.24, -0.24,  0.58, -0.26,  0.37,  0.58, -0.  , -0.  ],
       [ 0.58, -0.  ,  0.  ,  0.  , -0.  ,  0.  , -0.  ,  0.58,  0.58],
       [ 0.  , -0.35, -0.35,  0.16,  0.25, -0.8 ,  0.16, -0.  , -0.  ]])
```




```
U = [[문서1],
     [문서2],
     [문서3],
     [문서4]]
```

```
S = [[토픽1, 0, 0, 0],
     [0, 토픽2, 0, 0],
     [0, 0, 토픽3, 0],
     [0, 0, 0, 토픽4]]
```

잠재 의미 분석, LSA

▶ U : 문서-토픽 행렬

```
[18]  1 U.round(2)

array([[-0.24,  0.75,  0.  , -0.62],
       [-0.51,  0.44, -0.  ,  0.74],
       [-0.83, -0.49, -0.  , -0.27],
       [-0.  , -0.  ,  1.  ,  0.  ]])
```

4차원

⇔

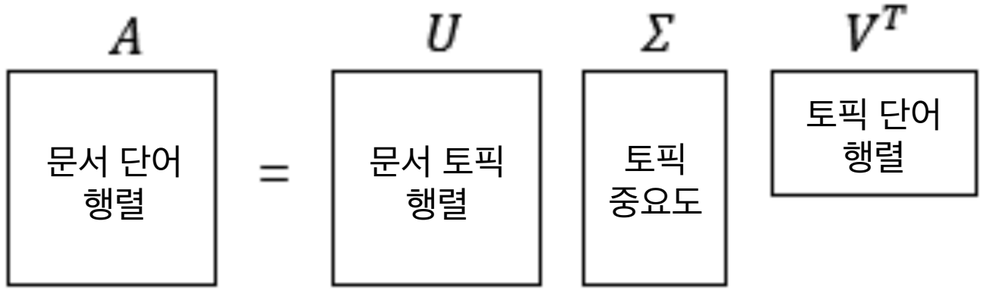
```
1 A

array([[0, 0, 0, 1, 0, 1, 1, 0, 0],
       [0, 0, 0, 1, 1, 0, 1, 0, 0],
       [0, 1, 1, 0, 2, 0, 0, 0, 0],
       [1, 0, 0, 0, 0, 0, 0, 1, 1]])
```

9차원

```
[19]  1 S.round(2)

array([[2.69, 0.  , 0.  , 0.  ],
       [0.  , 2.05, 0.  , 0.  ],
       [0.  , 0.  , 1.73, 0.  ],
       [0.  , 0.  , 0.  , 0.77]])
```

```
[20]  1 VT.round(2)

array([[-0.  , -0.31, -0.31, -0.28, -0.8 , -0.09, -0.28, -0.  , -0.  ],
       [ 0.  , -0.24, -0.24,  0.58, -0.26,  0.37,  0.58, -0.  , -0.  ],
       [ 0.58, -0.  ,  0.  ,  0.  , -0.  ,  0.  , -0.  ,  0.58,  0.58],
       [ 0.  , -0.35, -0.35,  0.16,  0.25, -0.8 ,  0.16, -0.  , -0.  ]])
```

Full SVD




```
U = [[문서1],
     [문서2],
     [문서3],
     [문서4]]
```

```
S = [[토픽1, 0, 0, 0],
     [0, 토픽2, 0, 0],
     [0, 0, 토픽3, 0],
     [0, 0, 0, 토픽4]]
```

▶ U의 행벡터는 문서의 저차원 표현

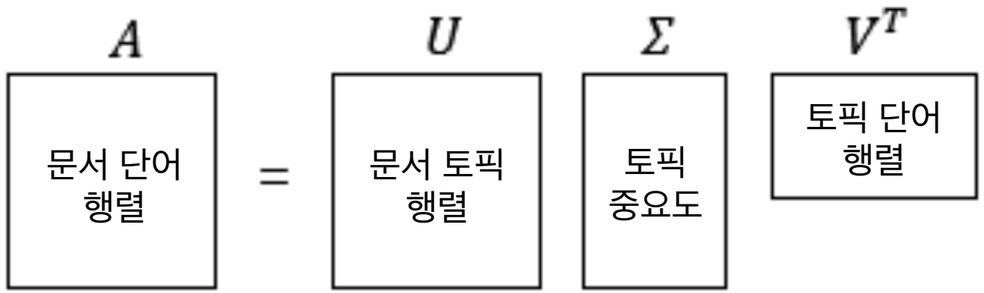
- ▶ 토픽 개수와 같은 차원의 벡터 표현일 뿐, 숫자들이 어떤 의미를 지니는지는 모름
- ▶ ex. 문서 1의 토픽 벡터의 1번째 숫자인 -0.24 = 문서 1에서 토픽 1의 중요도 ???

잠재 의미 분석, LSA

▶ V^T : 토픽-단어 행렬

```
[18]  1 U.round(2)

     array([[-0.24,  0.75,  0.  , -0.62],
            [-0.51,  0.44, -0.  ,  0.74],
            [-0.83, -0.49, -0.  , -0.27],
            [-0.  , -0.  ,  1.  ,  0.  ]])

[19]  1 S.round(2)

     array([[2.69, 0.  , 0.  , 0.  ],
            [0.  , 2.05, 0.  , 0.  ],
            [0.  , 0.  , 1.73, 0.  ],
            [0.  , 0.  , 0.  , 0.77]])

[20]  1 VT.round(2)

     array([[-0.  , -0.31, -0.31, -0.28, -0.8 , -0.09, -0.28, -0.  , -0.  ],
            [ 0.  , -0.24, -0.24,  0.58, -0.26,  0.37,  0.58, -0.  , -0.  ],
            [ 0.58, -0.  ,  0.  ,  0.  , -0.  ,  0.  , -0.  ,  0.58,  0.58],
            [ 0.  , -0.35, -0.35,  0.16,  0.25, -0.8 ,  0.16, -0.  , -0.  ]])
```

Full SVD




```
U = [[문서1],
     [문서2],
     [문서3],
     [문서4]]
```

```
S = [[토픽1, 0, 0, 0],
     [0, 토픽2, 0, 0],
     [0, 0, 토픽3, 0],
     [0, 0, 0, 토픽4]]
```

```
VT = [[토픽1],
      [토픽2],
      [토픽3],
      [토픽4]]
```

▶ V^T 의 행벡터는 토픽별 단어 벡터

▶ 단어 개수와 같은 차원의 벡터 표현일 뿐, 숫자들이 어떤 의미를 지니는지는 모름

잠재 의미 분석, LSA

▶ V^T : 토픽-단어 행렬

```
[18]  1 U.round(2)

      array([[-0.24,  0.75,  0.  , -0.62],
             [-0.51,  0.44, -0.  ,  0.74],
             [-0.83, -0.49, -0.  , -0.27],
             [-0.  , -0.  ,  1.  ,  0.  ]])

[19]  1 S.round(2)

      array([[2.69, 0.  , 0.  , 0.  ],
             [0.  , 2.05, 0.  , 0.  ],
             [0.  , 0.  , 1.73, 0.  ],
             [0.  , 0.  , 0.  , 0.77]])

[20]  1 VT.round(2)

      array([[-0.  , -0.31, -0.31, -0.28, -0.8 , -0.09, -0.28, -0.  , -0.  ],
             [ 0.  , -0.24, -0.24,  0.58, -0.26,  0.37,  0.58, -0.  , -0.  ],
             [ 0.58, -0.  ,  0.  ,  0.  , -0.  ,  0.  , -0.  ,  0.58,  0.58],
             [ 0.  , -0.35, -0.35,  0.16,  0.25, -0.8 ,  0.16, -0.  , -0.  ]])
```

-	과일이	길고	노란	먹고	바나나	사과	싶은	저는	좋아요
문서1	0	0	0	1	0	1	1	0	0
문서2	0	0	0	1	1	0	1	0	0
문서3	0	1	1	0	2	0	0	0	0
문서4	1	0	0	0	0	0	0	1	1

▶ V^T 의 열벡터들은 단어별 토픽 벡터.

▶ U에서와 마찬가지로, 토픽의 개수와 같은 차원으로(저차원) 압축된 표현

잠재 의미 분석, LSA

▶ truncated SVD를 이용한 LSA

```
[29]  1 U = U[:,:2]
      2
      3 U.round(2)
```

```
array([[-0.24,  0.75],
       [-0.51,  0.44],
       [-0.83, -0.49],
       [-0.  , -0.  ]])
```

```
[30]  1 s = s[:2]
      2 S = np.diag(s)
      3
      4 S.round(2)
```

```
array([[2.69, 0.  ],
       [0.  , 2.05]])
```

```
[31]  1 VT = VT[:2,:]
      2
      3 VT.round(2)
```

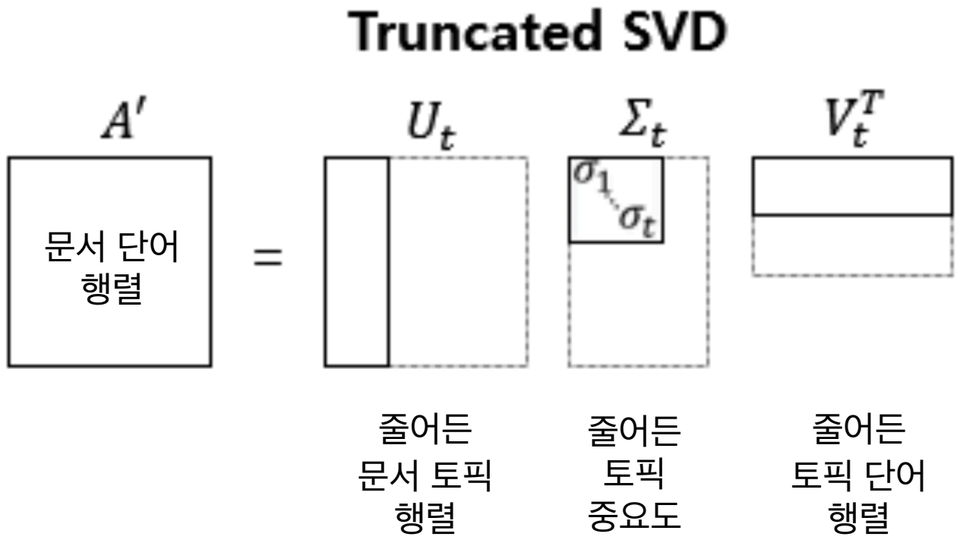
```
array([[-0.  , -0.31, -0.31, -0.28, -0.8 , -0.09, -0.28, -0.  , -0.  ],
       [ 0.  , -0.24, -0.24,  0.58, -0.26,  0.37,  0.58, -0.  , -0.  ]])
```

```
[32]  1 A_recon = U @ S @ VT
      2
      3 np.abs(A_recon.round(2))
```

```
array([[0.  , 0.17, 0.17, 1.08, 0.12, 0.62, 1.08, 0.  , 0.  ],
       [0.  , 0.2 , 0.2 , 0.91, 0.86, 0.45, 0.91, 0.  , 0.  ],
       [0.  , 0.93, 0.93, 0.03, 2.05, 0.17, 0.03, 0.  , 0.  ],
       [0.  , 0.  , 0.  , 0.  , 0.  , 0.  , 0.  , 0.  , 0.  ]])
```

```
[33]  1 A
```

```
array([[0, 0, 0, 1, 0, 1, 1, 0, 0],
       [0, 0, 0, 1, 1, 0, 1, 0, 0],
       [0, 1, 1, 0, 2, 0, 0, 0, 0],
       [1, 0, 0, 0, 0, 0, 0, 1, 1]])
```



잠재 의미 분석, LSA

▶ truncated SVD를 이용한 LSA

```
[29]  1 U = U[:,:2]
      2
      3 U.round(2)
```

```
array([[-0.24,  0.75],
       [-0.51,  0.44],
       [-0.83, -0.49],
       [-0.  , -0.  ]])
```

```
[30]  1 s = s[:2]
      2 S = np.diag(s)
      3
      4 S.round(2)
```

```
array([[2.69, 0.  ],
       [0.  , 2.05]])
```

```
[31]  1 VT = VT[:2,:]
      2
      3 VT.round(2)
```

```
array([[-0.  , -0.31, -0.31, -0.28, -0.8 , -0.09, -0.28, -0.  , -0.  ],
       [ 0.  , -0.24, -0.24,  0.58, -0.26,  0.37,  0.58, -0.  , -0.  ]])
```

-	과일이	길고	노란	먹고	바나나	사과	싶은	저는	좋아요
문서1	0	0	0	1	0	1	1	0	0
문서2	0	0	0	1	1	0	1	0	0
문서3	0	1	1	0	2	0	0	0	0
문서4	1	0	0	0	0	0	0	1	1

▶ 문서는 4개, 단어는 9개이지만 토픽은 2개

▶ 문서 표현 및 단어 표현이 2차원 벡터로 모두 압축된다

▶ 코사인 유사도로 문서/단어 비교 가능 + 추가 분석에 이용

잠재 의미 분석, LSA

▶ 질문 있으신가요?

```
[29]  1 U = U[:,:2]
      2
      3 U.round(2)
```

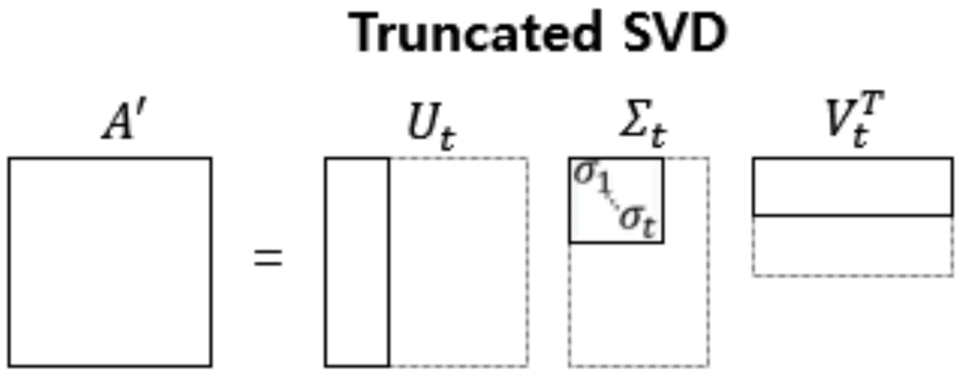
```
array([[-0.24,  0.75],
       [-0.51,  0.44],
       [-0.83, -0.49],
       [-0.  , -0.  ]])
```

```
[30]  1 s = s[:2]
      2 S = np.diag(s)
      3
      4 S.round(2)
```

```
array([[2.69, 0.  ],
       [0.  , 2.05]])
```

```
[31]  1 VT = VT[:2,:]
      2
      3 VT.round(2)
```

```
array([[-0.  , -0.31, -0.31, -0.28, -0.8 , -0.09, -0.28, -0.  , -0.  ],
       [ 0.  , -0.24, -0.24,  0.58, -0.26,  0.37,  0.58, -0.  , -0.  ]])
```



-	과일이	길고	노란	먹고	바나나	사과	싶은	저는	좋아요
문서1	0	0	0	1	0	1	1	0	0
문서2	0	0	0	1	1	0	1	0	0
문서3	0	1	1	0	2	0	0	0	0
문서4	1	0	0	0	0	0	0	1	1

잠재 디리클레 할당

Latent Dirichlet Allocation

Motivation

- LSA는 토픽모델링을 위한 기법이라기보다는 토픽을 고려한 문서 및 단어 표현 기법

  - 저차원 상의 벡터들의 의미를 해석하기 어려움

```
[29]  1 U = U[:,:2]
      2
      3 U.round(2)
```

```
array([[-0.24,  0.75],
       [-0.51,  0.44],
       [-0.83, -0.49],
       [-0.  , -0.  ]])
```

```
[30]  1 s = s[:2]
      2 S = np.diag(s)
      3
      4 S.round(2)
```

```
array([[2.69, 0.  ],
       [0.  , 2.05]])
```

```
[31]  1 VT = VT[:2,:]
      2
      3 VT.round(2)
```

```
array([[-0.  , -0.31, -0.31, -0.28, -0.8 , -0.09, -0.28, -0.  , -0.  ],
       [ 0.  , -0.24, -0.24,  0.58, -0.26,  0.37,  0.58, -0.  , -0.  ]])
```

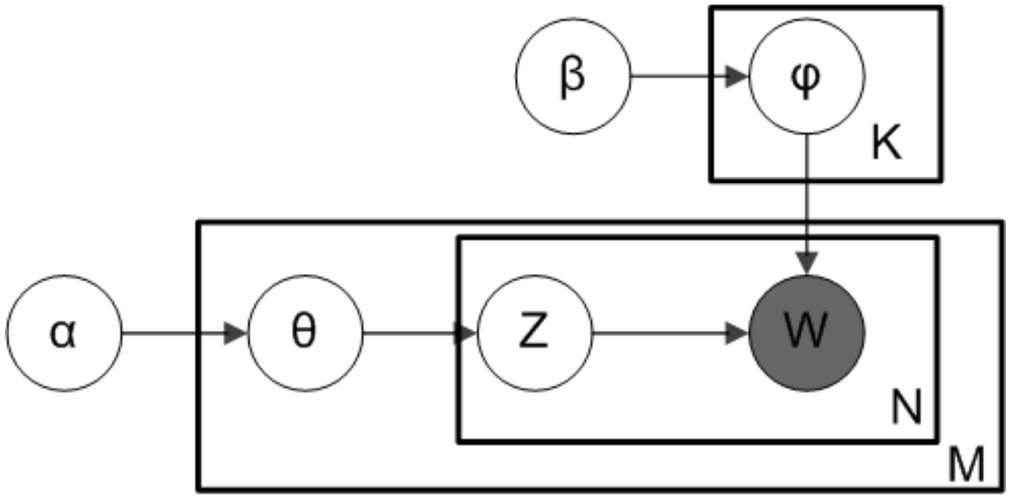
-	과일이	길고	노란	먹고	바나나	사과	싶은	저는	좋아요
문서1	0	0	0	1	0	1	1	0	0
문서2	0	0	0	1	1	0	1	0	0
문서3	0	1	1	0	2	0	0	0	0
문서4	1	0	0	0	0	0	0	1	1

각 원소는 문서별 단어 빈도수!

1번째 원소가 1번째 토픽에 대한 값
2번째 원소가 2번째 토픽에 대한 값
→ 그래서 무슨 값인데?

Motivation

▶ LDA는 토픽에 따라 다른 단어를 가진 문서가 생성되는 과정을 설명하는 확률 모델을 도입



LDA의 확률 모델

Graphical Model

▶ 변수들 간의 상호 의존 관계를 표현한 확률 모델

▶ $P(A, B, C)$

$$
\begin{aligned}
&P(A = 0, B = 0, C = 0)\\
&P(A = 0, B = 0, C = 1)\\
&P(A = 0, B = 1, C = 0)\\
&\dots\\
&P(A = 1, B = 1, C = 0)\\
&P(A = 1, B = 1, C = 1)
\end{aligned}
$$

$A,\ B,\ C$

Graphical Model

- 변수들 간의 상호 의존 관계를 표현한 확률 모델

  - $P(A, B, C) = P(A \mid C)P(C \mid B)P(B)$ (chain rule of probability)

P(B)

$$B \rightarrow C \rightarrow A$$

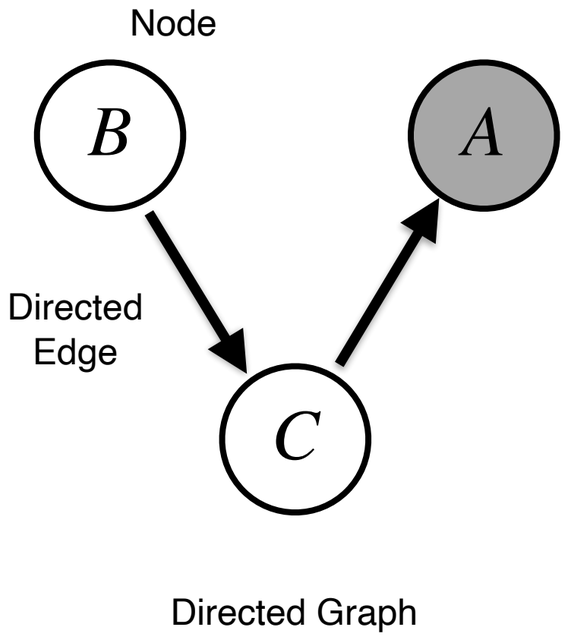
P(CIB) P(AIC)

Graphical Model

- 변수들 간의 상호 의존 관계를 표현한 확률 모델

- $P(A, B, C) = P(A \mid C)P(C \mid B)P(B)$

$$B \rightarrow C \rightarrow A$$



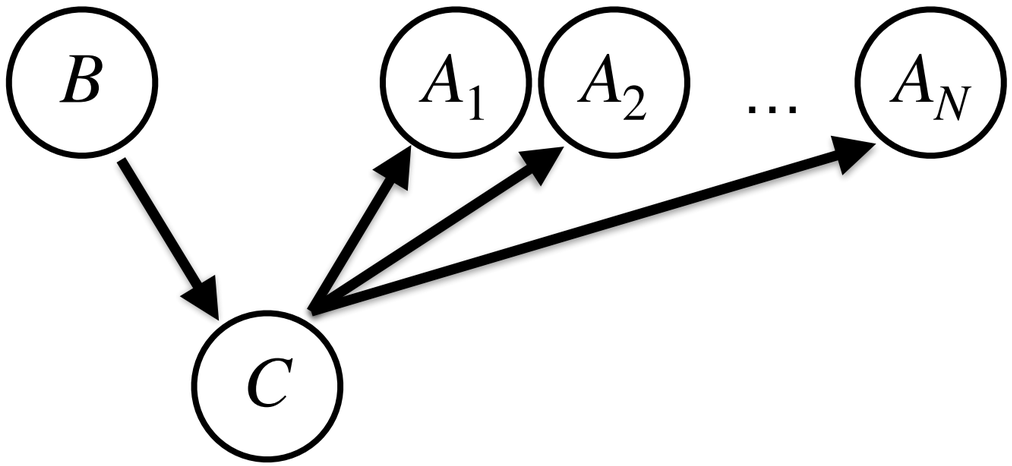
Directed Graph

Graphical Model

- 변수들 간의 상호 의존 관계를 표현한 확률 모델

- $P(A_1, A_2, \ldots, A_N, B, C) = P(A_1 \mid C)P(A_2 \mid C) \ldots P(A_N \mid C)P(C \mid B)P(B)$

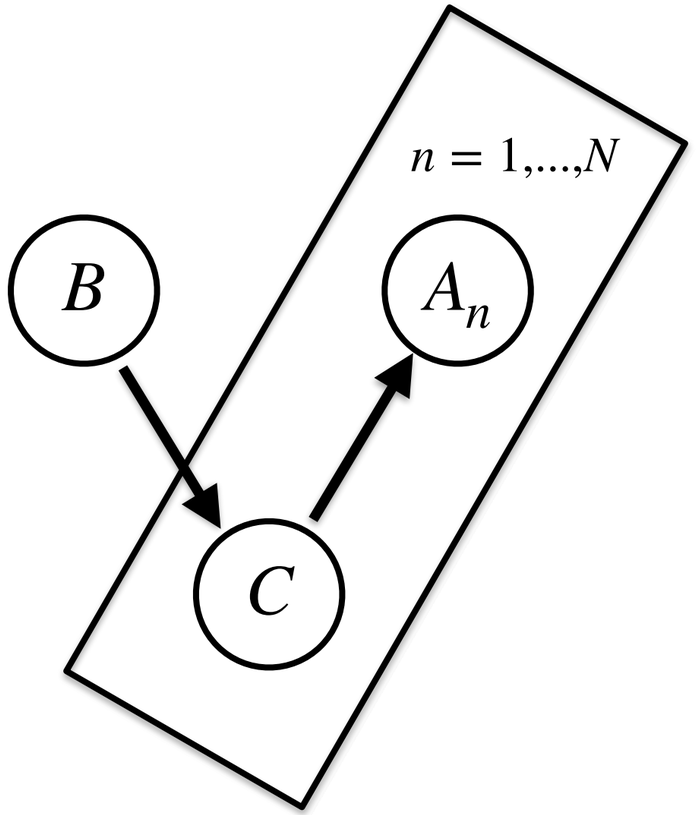
$$B \rightarrow C \rightarrow A_{n=1,2,\ldots,N}$$



Graphical Model

- 변수들 간의 상호 의존 관계를 표현한 확률 모델

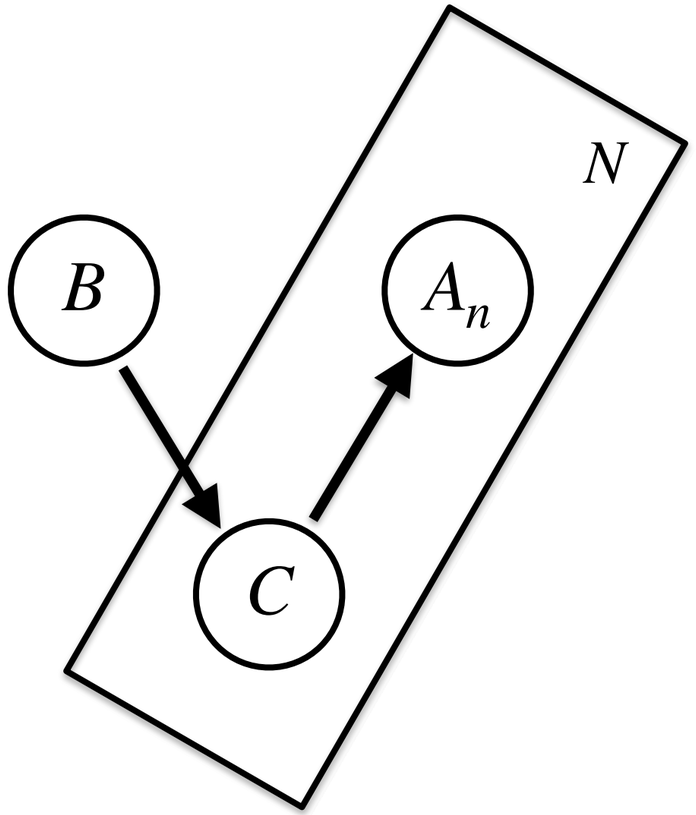
- $P(A_1, A_2, \ldots, A_N, B, C) = P(A_1 \mid C)P(A_2 \mid C) \ldots P(A_N \mid C)P(C \mid B)P(B)$



Graphical Model

- 변수들 간의 상호 의존 관계를 표현한 확률 모델

- $P(A_1, A_2, \ldots, A_N, B, C) = P(A_1 \mid C)P(A_2 \mid C) \ldots P(A_N \mid C)P(C \mid B)P(B)$



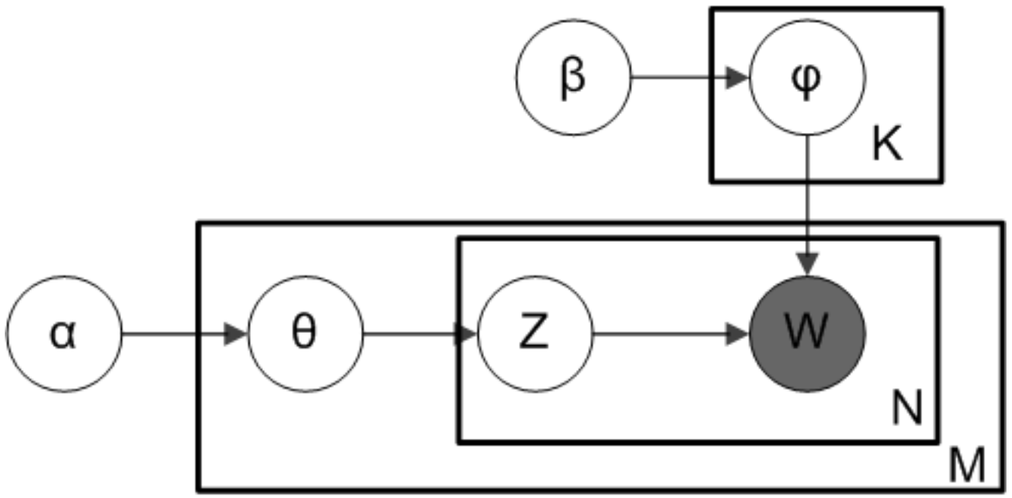
잠재 디리클레 할당

▶ **LDA**는 **토픽**에 따라 다른 **단어**를 가진 **문서**가 생성되는 과정을 설명하는 **그래피컬 모델**을 도입

▶ α, θ, Z : 문서 관련 변수

▶ β, ϕ : 토픽 관련 변수

▶ W : 단어 변수



LDA의 그래피컬 모델

잠재 디리클레 할당

▶ LDA의 원리

▶ “나는 이 **문서**를 작성하기 위해서 이런 **토픽**들을 넣을거고, 이런 **토픽**들을 위해서는 이런 **단어**들을 넣을 거야”

잠재 디리클레 할당

▶ LDA의 원리

▶ “나는 이 **문서**를 작성하기 위해서 이런 **토픽**들을 넣을거고, 이런 **토픽**들을 위해서는 이런 **단어**들을 넣을 거야”

문서_1 = 토픽_1 50% + 토픽_2 30% + 토픽_3 20%

문서_2 = 토픽_1 10% + 토픽_2 40% + 토픽_3 50%

토픽_1 = 단어_1 50% + 단어_2 30% + 단어_3 10% + 단어_4 10%

토픽_2 = 단어_1 20% + 단어_2 10% + 단어_3 40% + 단어_4 30%

토픽_3 = 단어_1 30% + 단어_2 40% + 단어_3 10% + 단어_4 20%

잠재 디리클레 할당

▶ LDA의 원리

▶ “나는 이 **문서**를 작성하기 위해서 이런 **토픽**들을 넣을거고, 이런 **토픽**들을 위해서는 이런 **단어**들을 넣을 거야”

문서_1 = 토픽_1 50% + 토픽_2 30% + 토픽_3 20%

문서_2 = 토픽_1 10% + 토픽_2 40% + 토픽_3 50%

토픽_1 = 단어_1 50% + 단어_2 30% + 단어_3 10% + 단어_4 10%

토픽_2 = 단어_1 20% + 단어_2 10% + 단어_3 40% + 단어_4 30%

토픽_3 = 단어_1 30% + 단어_2 40% + 단어_3 10% + 단어_4 20%

잠재 디리클레 할당

▶ LDA의 원리

▶ “나는 이 **문서**를 작성하기 위해서 이런 **토픽**들을 넣을거고, 이런 **토픽**들을 위해서는 이런 **단어**들을 넣을 거야”

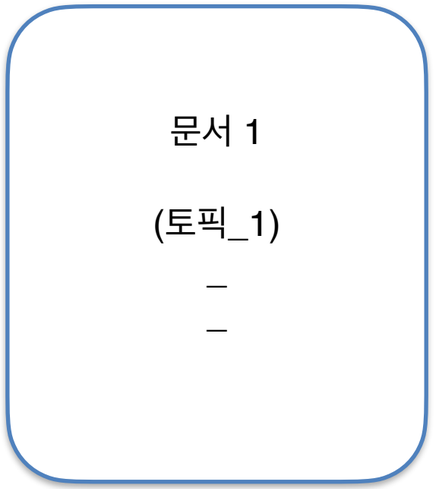
문서_1 = 토픽_1 50% + 토픽_2 30% + 토픽_3 20%

문서_2 = 토픽_1 10% + 토픽_2 40% + 토픽_3 50%

토픽_1 = 단어_1 50% + 단어_2 30% + 단어_3 10% + 단어_4 10%

토픽_2 = 단어_1 20% + 단어_2 10% + 단어_3 40% + 단어_4 30%

토픽_3 = 단어_1 30% + 단어_2 40% + 단어_3 10% + 단어_4 20%



잠재 디리클레 할당

▶ LDA의 원리

▶ “나는 이 **문서**를 작성하기 위해서 이런 **토픽**들을 넣을거고, 이런 **토픽**들을 위해서는 이런 **단어**들을 넣을 거야”

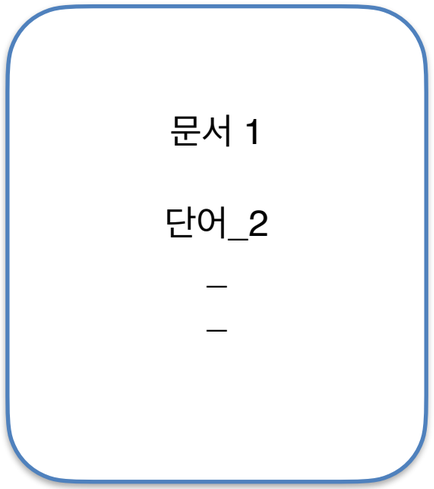
문서_1 = 토픽_1 50% + 토픽_2 30% + 토픽_3 20%

문서_2 = 토픽_1 10% + 토픽_2 40% + 토픽_3 50%

토픽_1 = 단어_1 50% + 단어_2 30% + 단어_3 10% + 단어_4 10%

토픽_2 = 단어_1 20% + 단어_2 10% + 단어_3 40% + 단어_4 30%

토픽_3 = 단어_1 30% + 단어_2 40% + 단어_3 10% + 단어_4 20%



잠재 디리클레 할당

▶ LDA의 원리

▶ “나는 이 **문서**를 작성하기 위해서 이런 **토픽**들을 넣을거고, 이런 **토픽**들을 위해서는 이런 **단어**들을 넣을 거야”

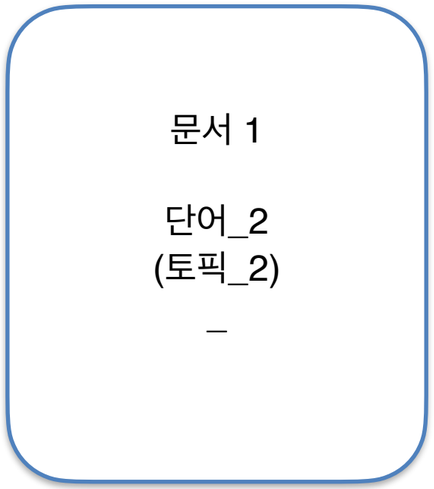
문서_1 = 토픽_1 50% + 토픽_2 30% + 토픽_3 20%

문서_2 = 토픽_1 10% + 토픽_2 40% + 토픽_3 50%

토픽_1 = 단어_1 50% + 단어_2 30% + 단어_3 10% + 단어_4 10%

토픽_2 = 단어_1 20% + 단어_2 10% + 단어_3 40% + 단어_4 30%

토픽_3 = 단어_1 30% + 단어_2 40% + 단어_3 10% + 단어_4 20%



잠재 디리클레 할당

▶ LDA의 원리

▶ “나는 이 **문서**를 작성하기 위해서 이런 **토픽**들을 넣을거고, 이런 **토픽**들을 위해서는 이런 **단어**들을 넣을 거야”

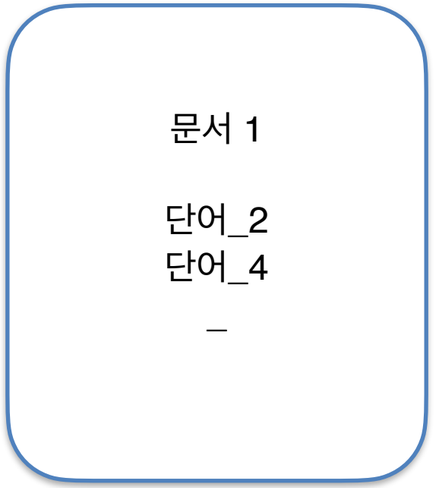
문서_1 = 토픽_1 50% + 토픽_2 30% + 토픽_3 20%

문서_2 = 토픽_1 10% + 토픽_2 40% + 토픽_3 50%

토픽_1 = 단어_1 50% + 단어_2 30% + 단어_3 10% + 단어_4 10%

토픽_2 = 단어_1 20% + 단어_2 10% + 단어_3 40% + 단어_4 30%

토픽_3 = 단어_1 30% + 단어_2 40% + 단어_3 10% + 단어_4 20%



잠재 디리클레 할당

▶ LDA의 원리

▶ “나는 이 **문서**를 작성하기 위해서 이런 **토픽**들을 넣을거고, 이런 **토픽**들을 위해서는 이런 **단어**들을 넣을 거야”

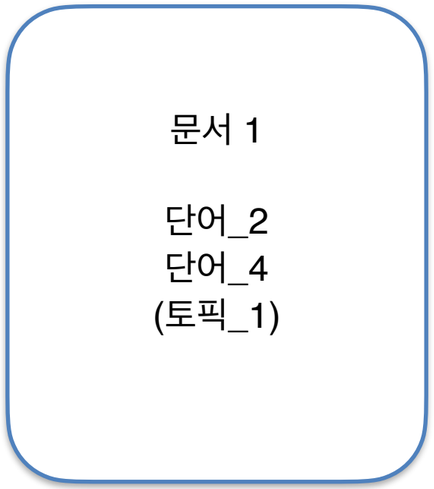
문서_1 = 토픽_1 50% + 토픽_2 30% + 토픽_3 20%

문서_2 = 토픽_1 10% + 토픽_2 40% + 토픽_3 50%

토픽_1 = 단어_1 50% + 단어_2 30% + 단어_3 10% + 단어_4 10%

토픽_2 = 단어_1 20% + 단어_2 10% + 단어_3 40% + 단어_4 30%

토픽_3 = 단어_1 30% + 단어_2 40% + 단어_3 10% + 단어_4 20%



잠재 디리클레 할당

▶ LDA의 원리

▶ “나는 이 **문서**를 작성하기 위해서 이런 **토픽**들을 넣을거고, 이런 **토픽**들을 위해서는 이런 **단어**들을 넣을 거야”

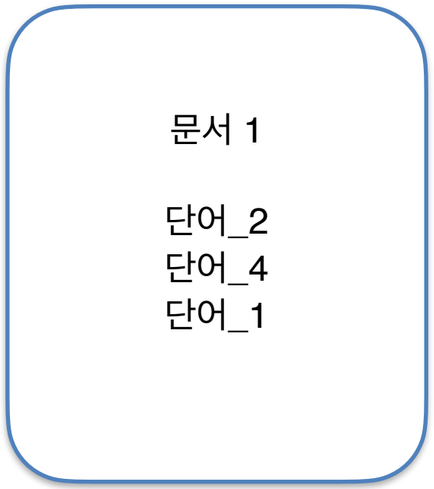
문서_1 = 토픽_1 50% + 토픽_2 30% + 토픽_3 20%

문서_2 = 토픽_1 10% + 토픽_2 40% + 토픽_3 50%

토픽_1 = 단어_1 50% + 단어_2 30% + 단어_3 10% + 단어_4 10%

토픽_2 = 단어_1 20% + 단어_2 10% + 단어_3 40% + 단어_4 30%

토픽_3 = 단어_1 30% + 단어_2 40% + 단어_3 10% + 단어_4 20%



잠재 디리클레 할당

▶ LDA의 원리

▶ “나는 이 **문서** D 를 작성하기 위해서 이런 **토픽** T 들을 넣을거고, 이런 **토픽**들을 위해서는 이런 **단어** W들을 넣을 거야”

문서_1 = 토픽_1 $P(T=1|D=1)$ + 토픽_2 $P(T=2|D=1)$+ … + 토픽_K $P(T=K|D=1)$

문서_2 = 토픽_1 $P(T=1|D=2)$ + 토픽_2 $P(T=2|D=2)$+ … + 토픽_K $P(T=K|D=2)$

…

문서_M = 토픽_1 $P(T=1|D=M)$ + 토픽_2 $P(T=2|D=M)$+ … + 토픽_K $P(T=K|D=M)$

토픽_1 = 단어_1 $P(W=1|T=1)$ + 단어_2 $P(W=2|T=1)$ + … + 단어_N $P(W=N|T=1)$

토픽_2 = 단어_1 $P(W=1|T=2)$ + 단어_2 $P(W=2|T=2)$ + … + 단어_N $P(W=N|T=2)$

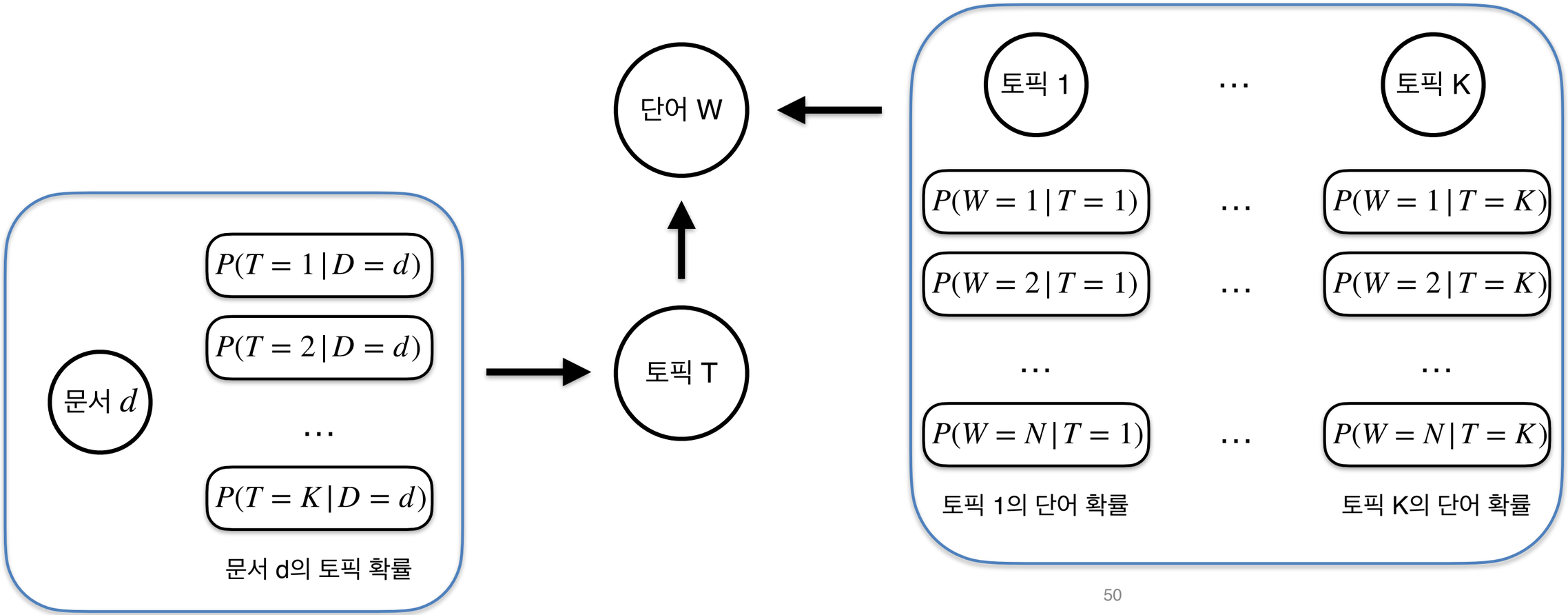
…

토픽_K = 단어_1 $P(W=1|T=K)$ + 단어_2 $P(W=2|T=K)$ + … + 단어_N $P(W=N|T=K)$

잠재 디리클레 할당

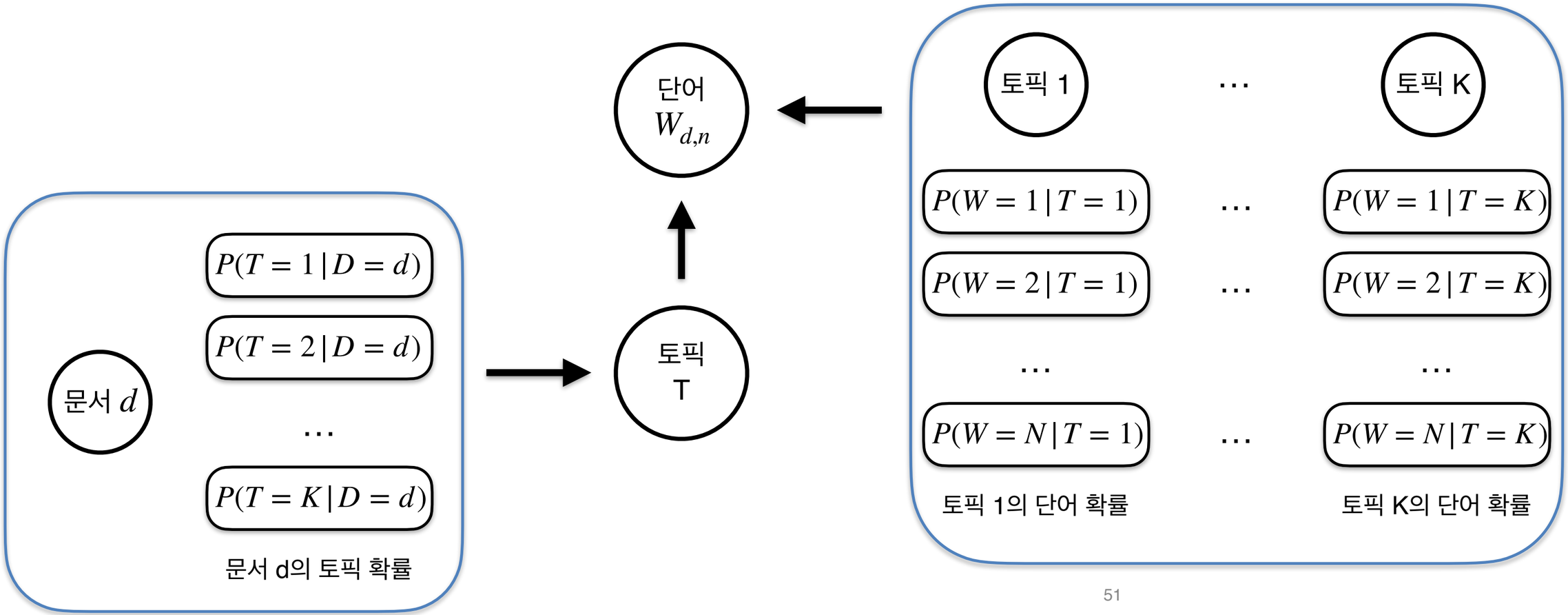
▶ LDA의 그래피컬 모델

▶ 문서에서 토픽이 뽑히고, 해당하는 토픽의 단어 확률에 따라 단어가 뽑힌다.



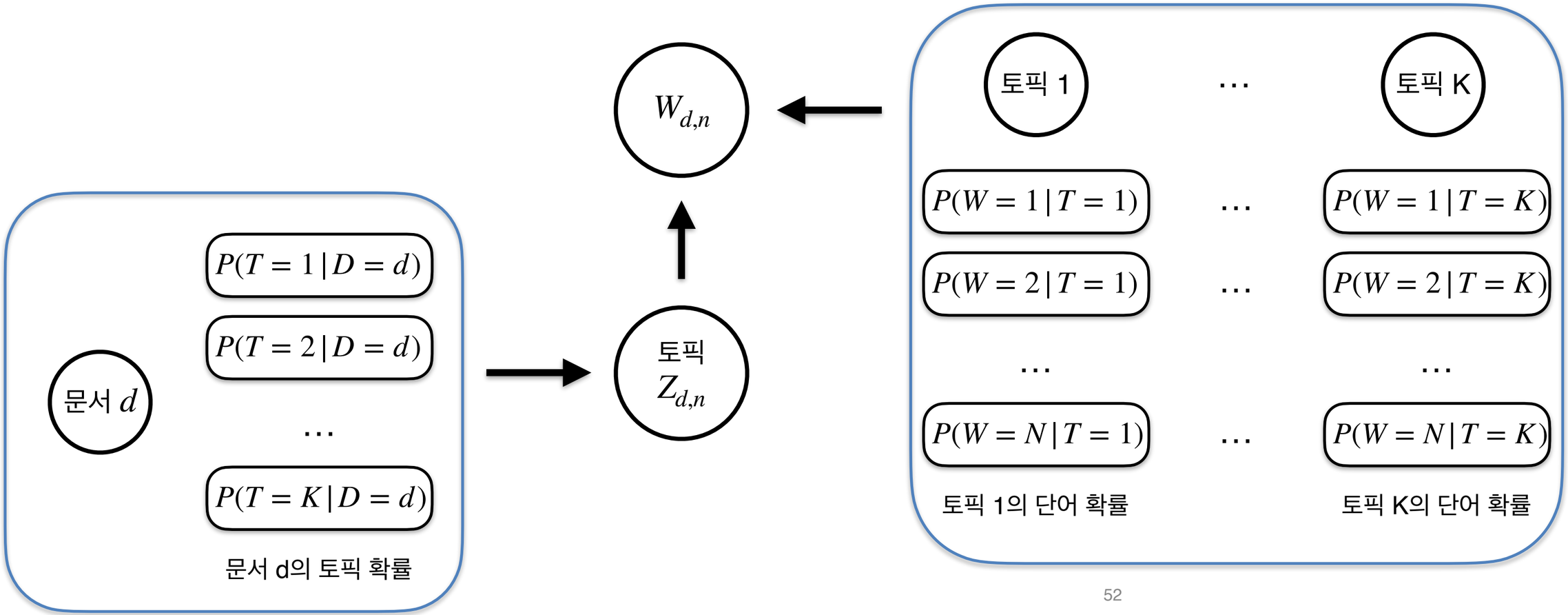
잠재 디리클레 할당

▶ $W_{d,n}$: 문서 d의 n번째 단어 ex) 문서3 의 2번째 단어가 BoW에서 5번째 단어라면, $W_{3,2} = 5$



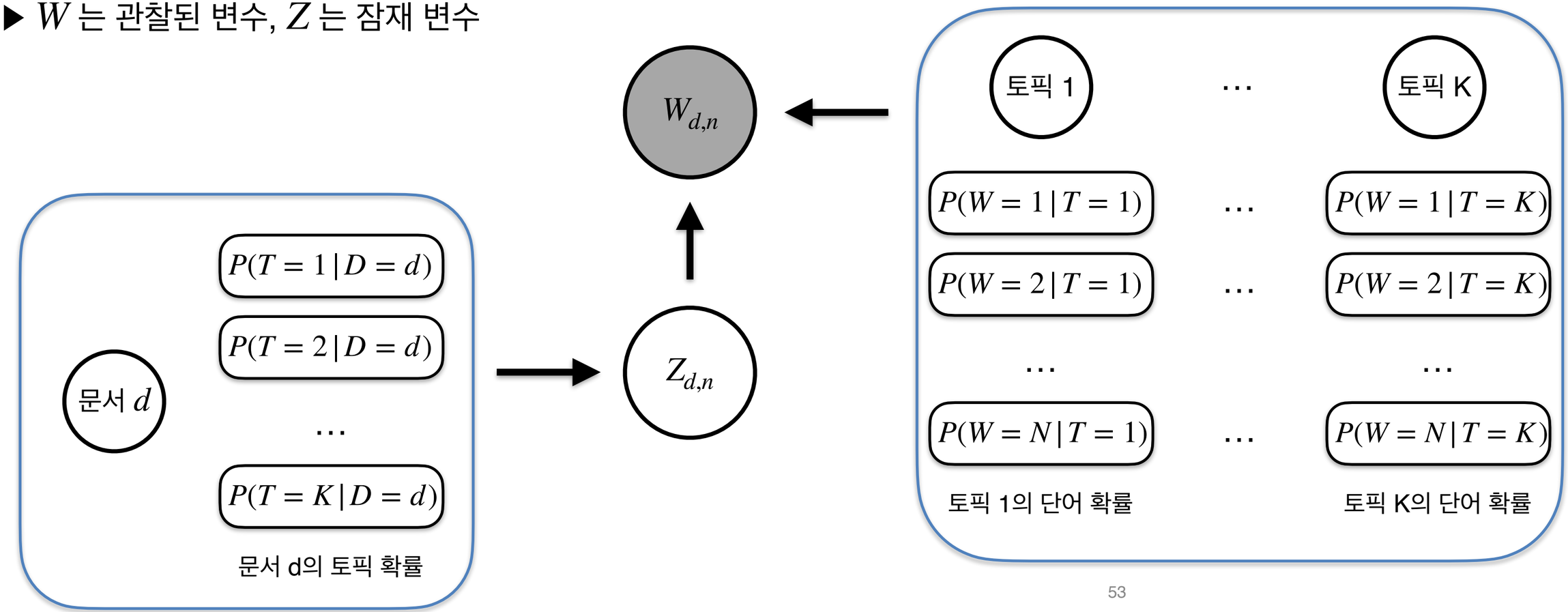
잠재 디리클레 할당

- $W_{d,n}$: 문서 d의 n번째 단어
- $Z_{d,n}$: 문서 d의 n번째 단어에 대한 토픽 ex) 문서3 의 2번째 단어가 1번째 토픽에서 왔다면, $Z_{3,2} = 1$



잠재 디리클레 할당

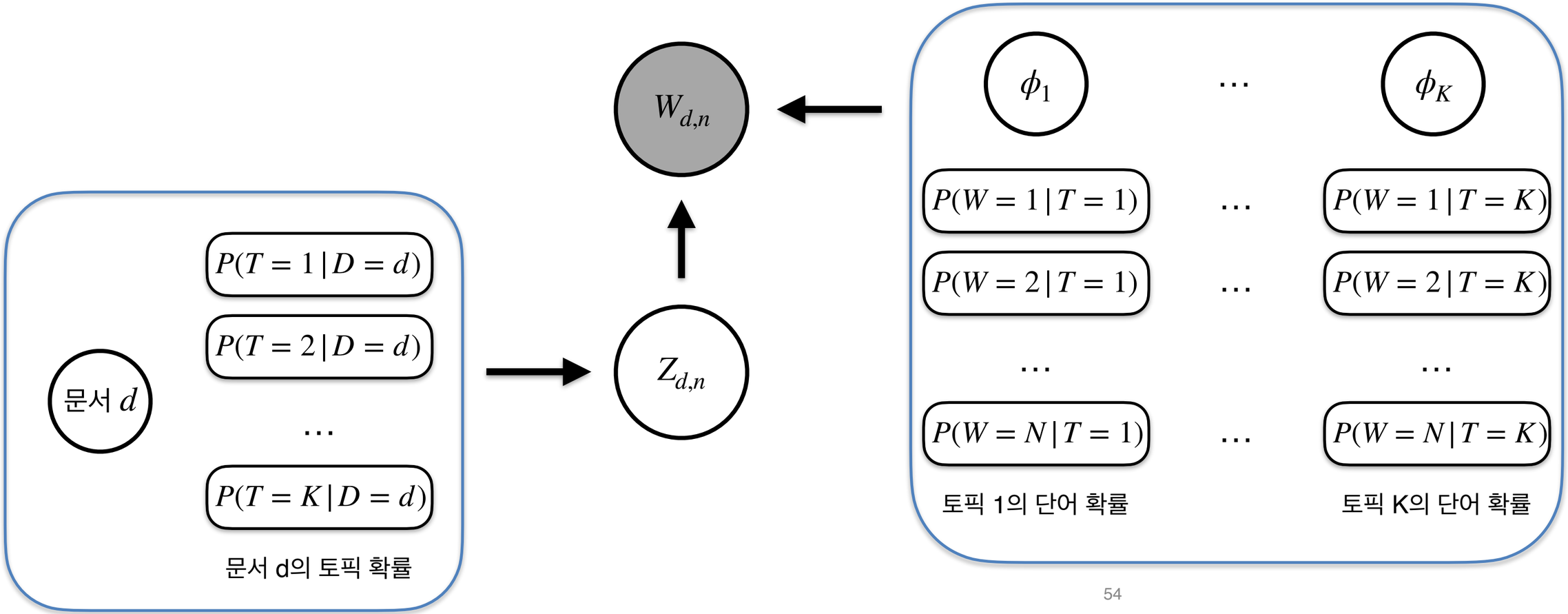
- $W_{d,n}$: 문서 d의 n번째 단어
- $Z_{d,n}$: 문서 d의 n번째 단어에 대한 토픽
- W 는 관찰된 변수, Z 는 잠재 변수



잠재 디리클레 할당

▶ ϕ_k : 토픽 k의 단어 확률을 벡터로 표현한 것.

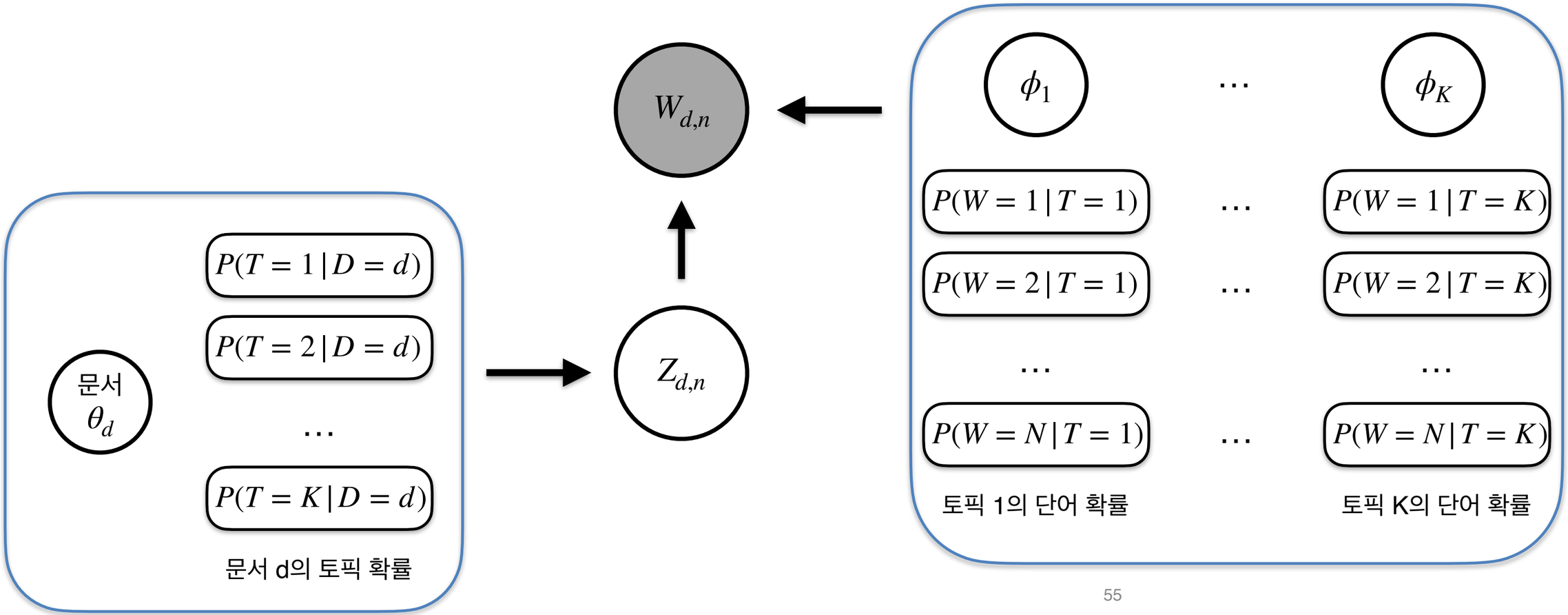
▶ ex) 토픽 1 = 단어 1 20% + 단어 2 40% + 단어 3 40% 라면, $\phi_1 = (0.2, 0.4, 0.4)$



잠재 디리클레 할당

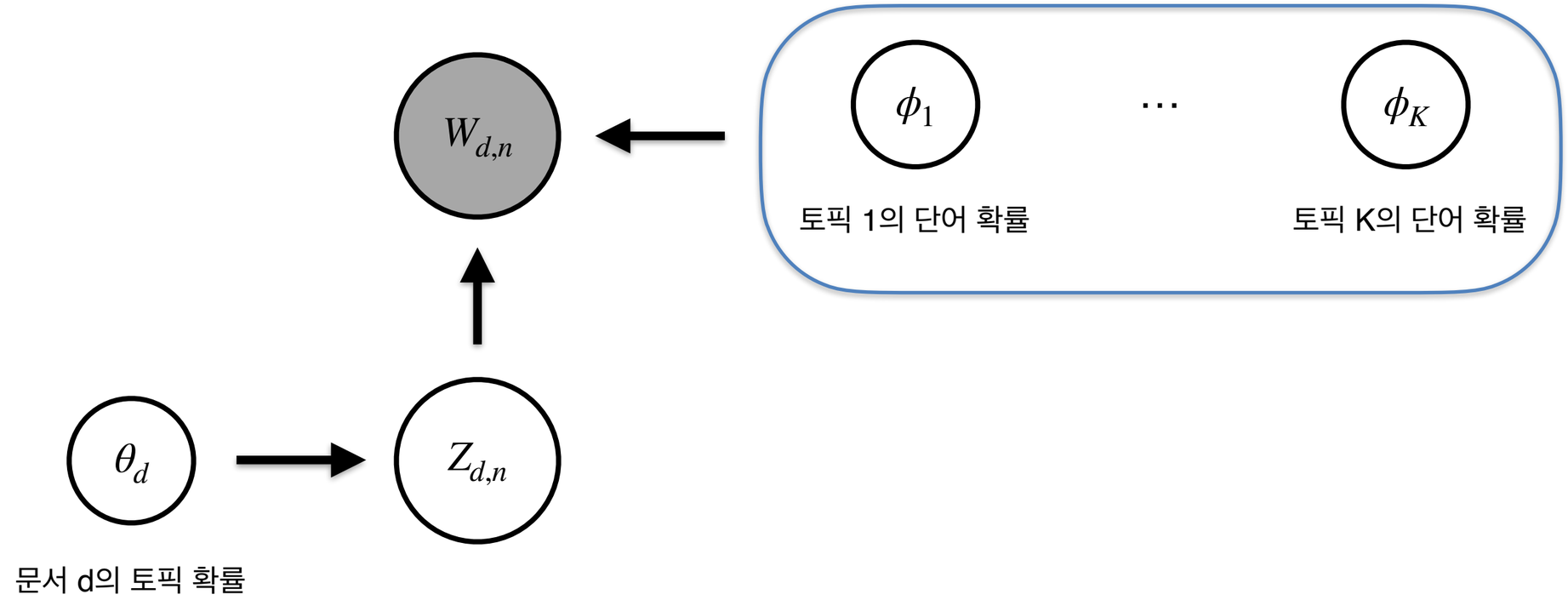
▶ θ_d : 문서 d의 토픽 확률을 벡터로 표현한 것.

▶ ex) 문서 2 = 토픽 1 70% + 토픽 2 30% 라면, $\theta_2 = (0.7, 0.3)$



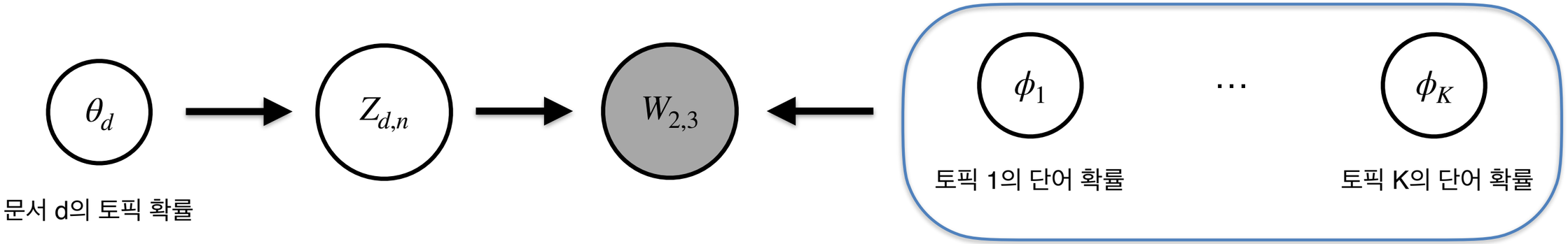
잠재 디리클레 할당

- ϕ_k : 토픽 k의 단어 확률
- θ_d : 문서 d의 토픽 확률



잠재 디리클레 할당

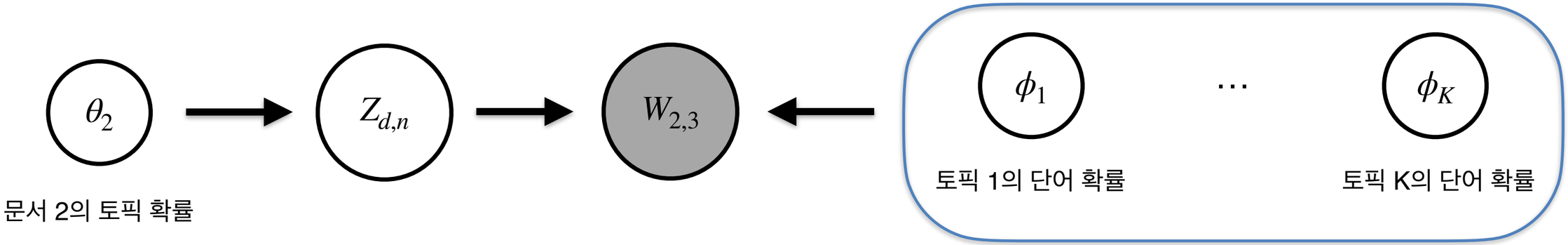
▶ LDA의 그래피컬 모델



▶ $W_{2,3}$: 문서 2의 3번째 단어

잠재 디리클레 할당

▶ LDA의 그래피컬 모델

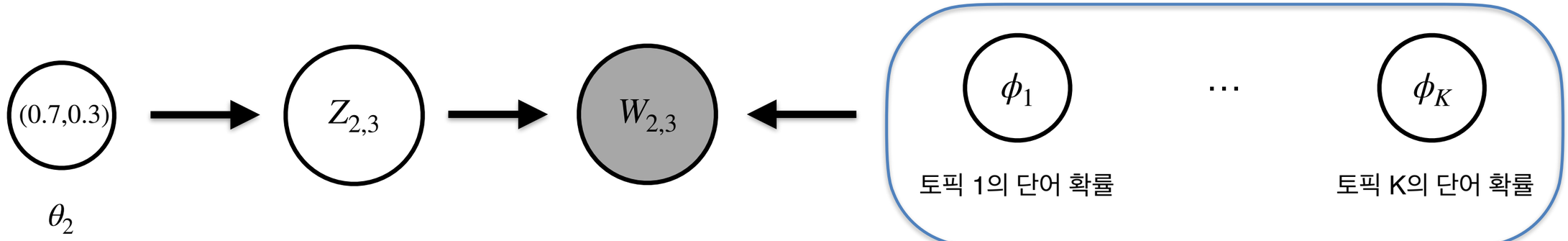


▶ 문서 2 = 토픽 1 70% + 토픽 2 30%

▶ $\theta_2 = (0.7, 0.3)$

잠재 디리클레 할당

▶ LDA의 그래피컬 모델



▶ 문서 2 = 토픽 1 70% + 토픽 2 30%

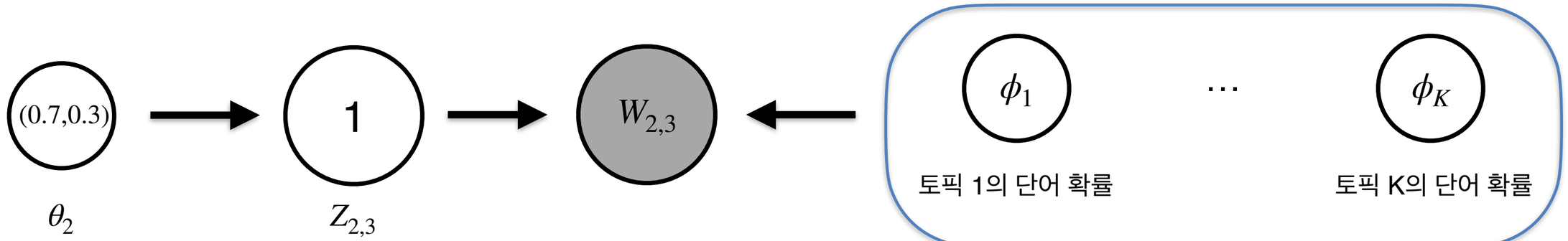
▶ $\theta_2 = (0.7,0.3)$

▶ 문서 2의 3번째 단어에 대한 토픽을 뽑는다.

▶ $Z_{2,3} \sim \text{Multinomial}(0.7,0.3) \rightarrow Z_{2,3} = 1$

잠재 디리클레 할당

▶ LDA의 그래피컬 모델



▶ 문서 2 = 토픽 1 70% + 토픽 2 30%

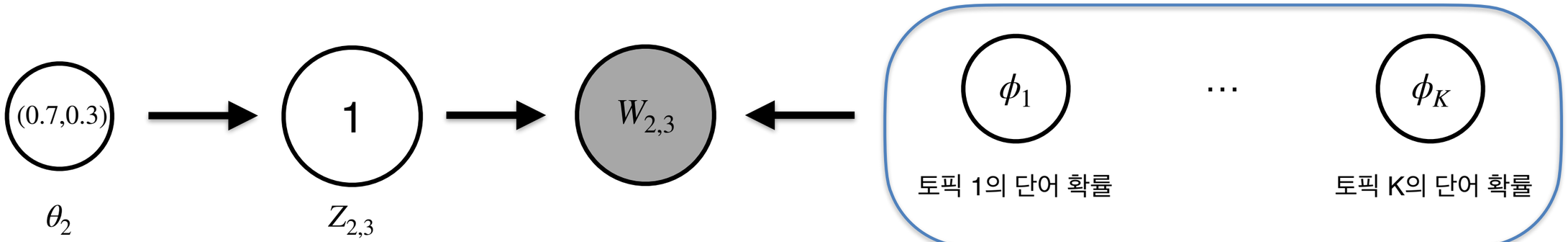
▶ $\theta_2 = (0.7,0.3)$

▶ 문서 2의 3번째 단어에 대한 토픽을 뽑는다.

▶ $Z_{2,3} \sim \text{Multinomial}(0.7,0.3) \rightarrow Z_{2,3} = 1$

잠재 디리클레 할당

▶ LDA의 그래피컬 모델



▶ 문서 2 = 토픽 1 70% + 토픽 2 30%

▶ $\theta_2 = (0.7,0.3)$

▶ 문서 2의 3번째 단어에 대한 토픽을 뽑는다.

▶ $Z_{2,3} \sim \text{Multinomial}(0.7,0.3) \rightarrow Z_{2,3} = 1$

▶ 토픽 1 = 단어 1 20% + 단어 2 40% + 단어 3 40%

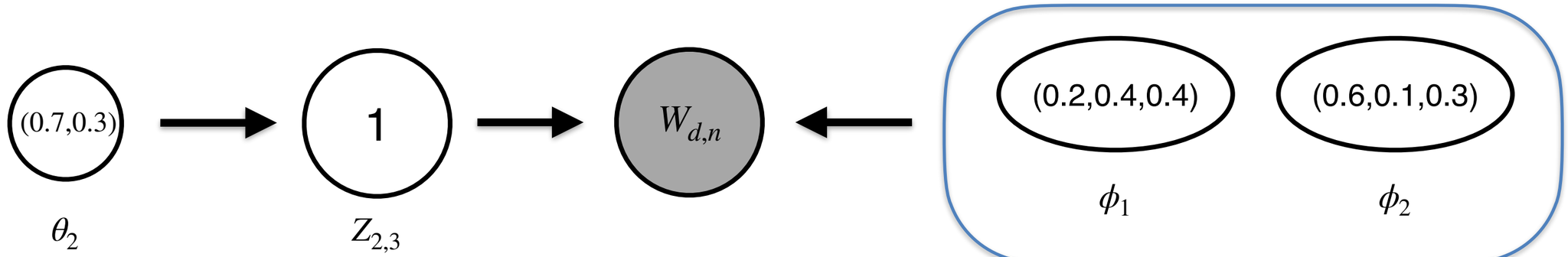
▶ $\phi_1 = (0.2,0.4,0.4)$

▶ 토픽 2 = 단어 1 60% + 단어 2 10% + 단어 3 30%

▶ $\phi_2 = (0.6,0.1,0.3)$

잠재 디리클레 할당

▶ LDA의 그래피컬 모델



▶ 문서 2 = 토픽 1 70% + 토픽 2 30%

▶ $\theta_2 = (0.7,0.3)$

▶ 문서 2의 3번째 단어에 대한 토픽을 뽑는다.

▶ $Z_{2,3} \sim \text{Multinomial}(0.7,0.3) \rightarrow Z_{2,3} = 1$

▶ 토픽 1 = 단어 1 20% + 단어 2 40% + 단어 3 40%

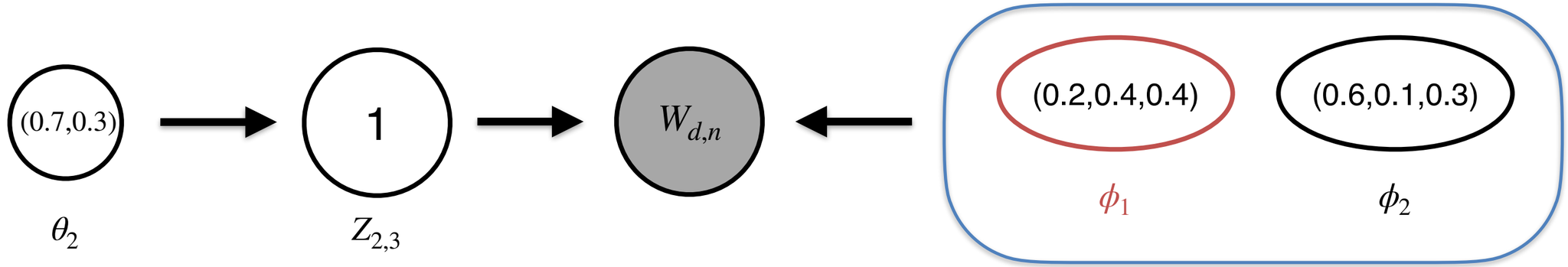
▶ $\phi_1 = (0.2,0.4,0.4)$

▶ 토픽 2 = 단어 1 60% + 단어 2 10% + 단어 3 30%

▶ $\phi_2 = (0.6,0.1,0.3)$

잠재 디리클레 할당

▶ LDA의 그래피컬 모델

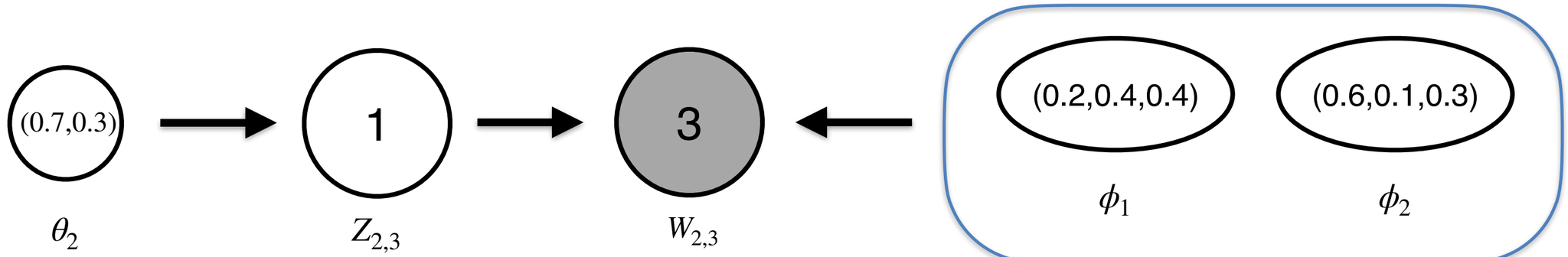


▶ 선택된 토픽 $Z_{2,3} = 1$에 해당하는 토픽의 확률분포 $\phi_{Z_{2,3}} = \phi_1$ 에서 단어 $W_{2,3}$ 을 뽑는다.

▶ $W_{2,3} \sim \text{Multinomial}(0.2,0.4,0.4) \rightarrow W_{2,3} = 3$

잠재 디리클레 할당

▶ LDA의 그래피컬 모델



▶ 뽑힌 토픽 $Z_{2,3}$에 해당하는 토픽의 확률분포 $\phi_{Z_{2,3}} = \phi_1$ 에서 단어 $W_{2,3}$ 을 뽑는다.

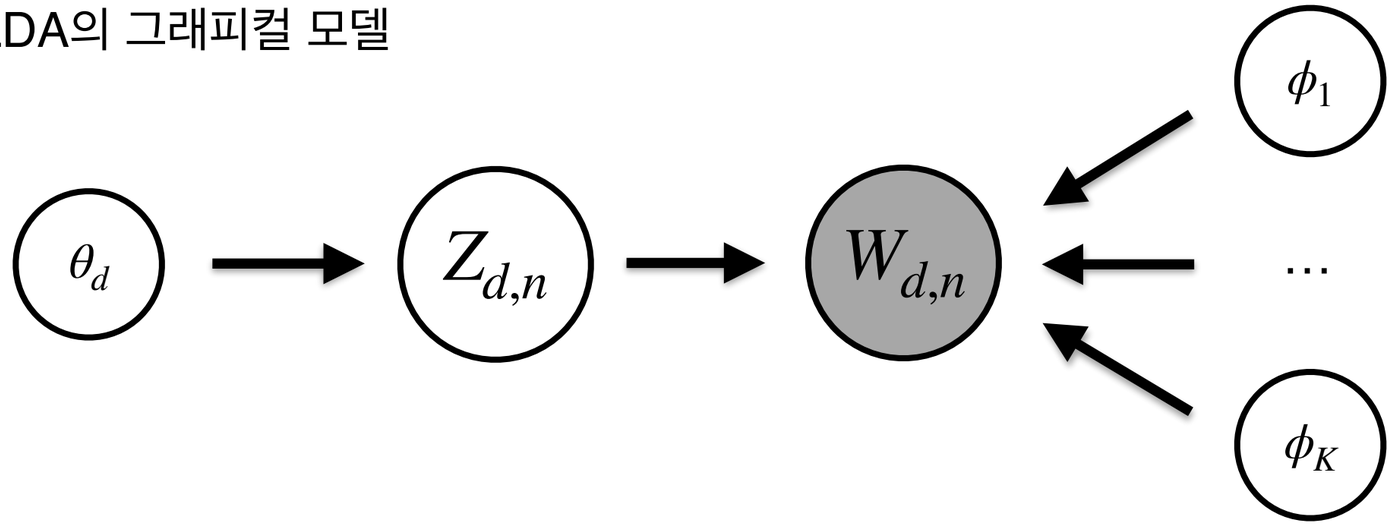
▶ $W_{2,3} \sim \text{Multinomial}(0.2,0.4,0.4) \rightarrow W_{2,3} = 3$

▶ Bag-of-Words(BoW) 에서 3번째 단어를 가져온 것이 실제 문서2의 3번째로 생성된 단어가 된다.

-	과일이	길고	노란	먹고	바나나	사과	싶은	저는	좋아요
문서1	0	0	0	1	0	1	1	0	0
문서2	0	0	0	1	1	0	1	0	0
문서3	0	1	1	0	2	0	0	0	0
문서4	1	0	0	0	0	0	0	1	1

잠재 디리클레 할당

▶ LDA의 그래피컬 모델



▶ 문서별 토픽 확률 θ_d

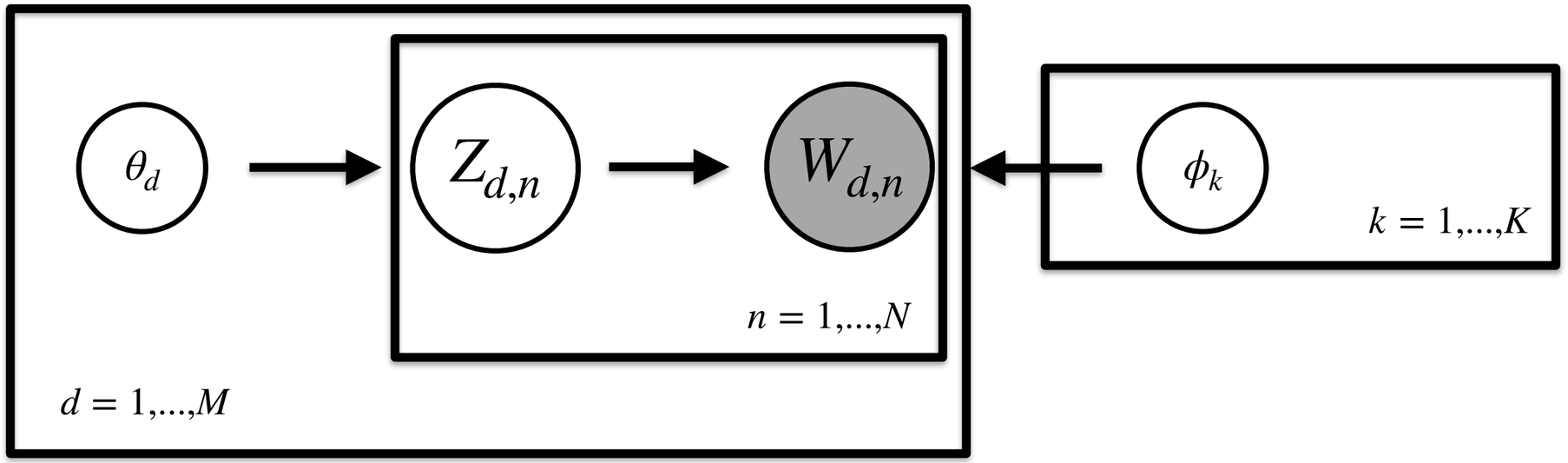
▶ 토픽별 단어 확률 $\phi_1, \ldots, \phi_K$

▶ 토픽 $Z_{d,n} \sim$ Multinomial(θ_d)

▶ 단어 $W_{d,n} \sim$ Multinomial($\phi_{Z_{d,n}}$)

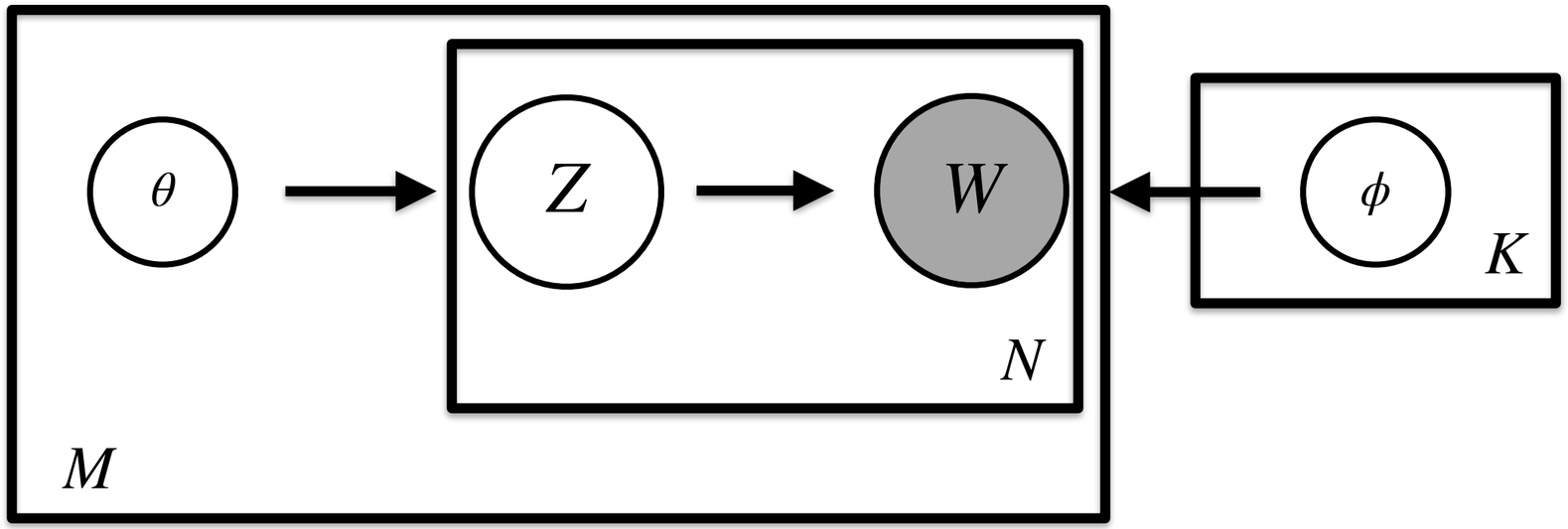
잠재 디리클레 할당

▶ LDA의 그래피컬 모델



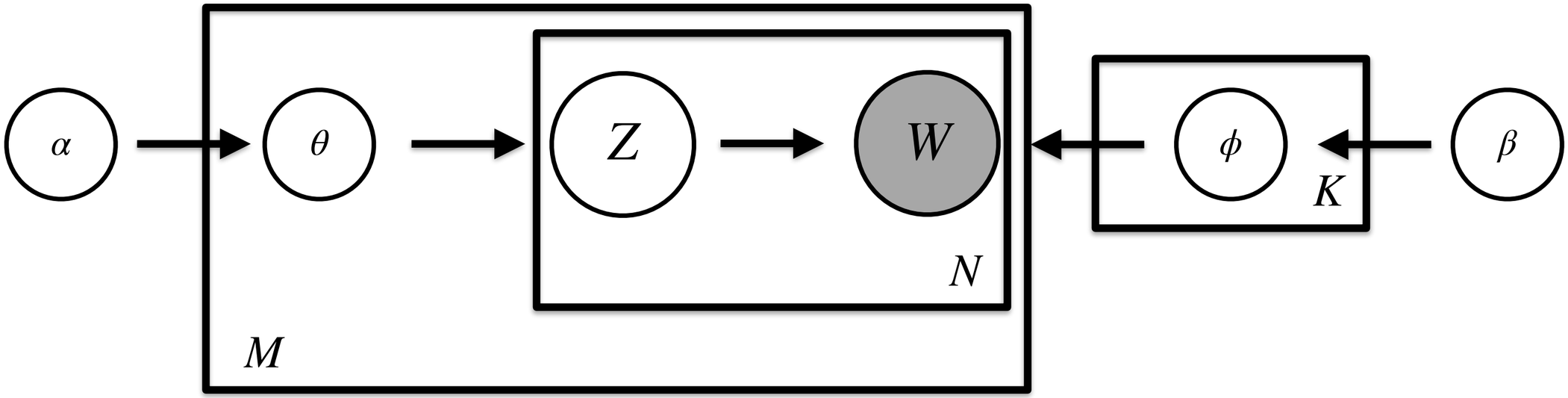
잠재 디리클레 할당

▶ LDA의 그래피컬 모델



잠재 디리클레 할당

▶ LDA의 그래피컬 모델



$$\theta \sim \text{Dirichlet}(\alpha)$$

$$\phi \sim \text{Dirichlet}(\beta)$$

LDA 실습 : 뉴스 헤드라인 데이터

▶ 뉴스 헤드라인 데이터

```
[ ]  1 import pandas as pd
     2 import urllib.request
     3
     4 urllib.request.urlretrieve("https://raw.githubusercontent.com/franciscadias/data/master/abcnews-date-text.csv", filename="abcnews-date-text.csv")
     5 data = pd.read_csv('abcnews-date-text.csv', error_bad_lines=False)
```

	publish_date	headline_text
0	20030219	aba decides against community broadcasting lic...
1	20030219	act fire witnesses must be aware of defamation
2	20030219	a g calls for infrastructure protection summit
3	20030219	air nz staff in aust strike for pay rise
4	20030219	air nz strike to affect australian travellers

```
[ ]  1 len(data)
```

```
1082168
```

LDA 실습 : 뉴스 헤드라인 데이터

▶ 분석을 위해 제목만 뽑아낸 뒤, 텍스트 전처리를 수행합니다.

```
[ ]  1 text = data[["headline_text"]]
```

```
[ ]  1 text.head()
```

	headline_text
0	aba decides against community broadcasting lic...
1	act fire witnesses must be aware of defamation
2	a g calls for infrastructure protection summit
3	air nz staff in aust strike for pay rise
4	air nz strike to affect australian travellers

▶ 불용어 제거 : 토픽 분석에 크게 도움이 되지 않는 against, be, of, a, in, to 등의 be 동사 및 전치사를 제거합니다.

▶ 표제어 추출 : 3인칭 단수 표현을 1인칭으로 바꾸고, 과거 현재형 동사를 현재형으로 바꿉니다.

▶ 짧은 단어 제거 : 길이가 3 이하인 단어 제거

▶ 단어 토큰화 : nltk.word_tokenize 를 이용해서 텍스트를 원핫벡터로 변환

LDA 실습 : 뉴스 헤드라인 데이터

▶ 전처리가 끝난 문서 단어 행렬로부터 TF-IDF 행렬을 준비합니다.

```
[ ]  1 from sklearn.feature_extraction.text import TfidfVectorizer
     2
     3 vectorizer = TfidfVectorizer(stop_words='english', max_features= 1000)
     4 X = vectorizer.fit_transform(text['headline_text'])
     5
     6 X.shape
```

```
(1082168, 1000)
```

▶ sklearn.decomposition의 LatentDirichletAllocation 클래스로 LDA를 수행합니다.

```
[ ]  1 from sklearn.decomposition import LatentDirichletAllocation
     2
     3 lda_model=LatentDirichletAllocation(n_components=10, learning_method='online', random_state=777, max_iter=1)
```

```
[ ]  1 lda_top=lda_model.fit_transform(X)
```

LDA 실습 : 뉴스 헤드라인 데이터

▶ LDA 결과는 LatentDirichletAllocation 클래스의 components_ 인자를 통해 볼 수 있습니다.

▶ components_는 각 토픽의 단어 확률을 보여줍니다.

```
[ ]   1 lda_model.components_.shape
```

```
(10, 1000)
```

```
[ ]   1 lda_model.components_[:4,:8].round(0)
```

```
array([[   0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.],
       [   0.,    0., 1440.,    0.,    0.,    0.,    0.,    0.],
       [1830.,  872.,    0.,    0.,    0., 1310.,    0.,    0.],
       [   0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.]])
```

LDA 실습 : 뉴스 헤드라인 데이터

▶ 각 토픽별로 큰 확률을 지닌 단어들을 보고, 어떤 토픽인지 유추할 수 있습니다.

```
[ ]  1 def get_topics_clean(components, feature_names, n=5):
     2     for idx, topic in enumerate(components):
     3         print("Topic %d:" % (idx+1), ", ".join([feature_names[i] for i in topic.argsort()[:-n - 1:-1]]))
     4
     5 get_topics_clean(lda_model.components_, terms)
```

```
Topic 1: police, coast, year, state, life
Topic 2: north, home, open, west, water
Topic 3: new, world, death, country, dies
Topic 4: australia, china, hospital, say, hour
Topic 5: wa, sydney, south, perth, canberra
Topic 6: man, melbourne, court, charged, murder
Topic 7: says, queensland, day, attack, brisbane
Topic 8: government, sa, adelaide, 2016, tasmanian
Topic 9: trump, election, nsw, rural, win
Topic 10: australian, nt, league, test, big
```

LDA 실습 : 뉴스 헤드라인 데이터

▶ 마지막으로, 문서들이 어떤 토픽으로 이루어져 있는지를 확인함으로써 LDA가 완료됩니다.

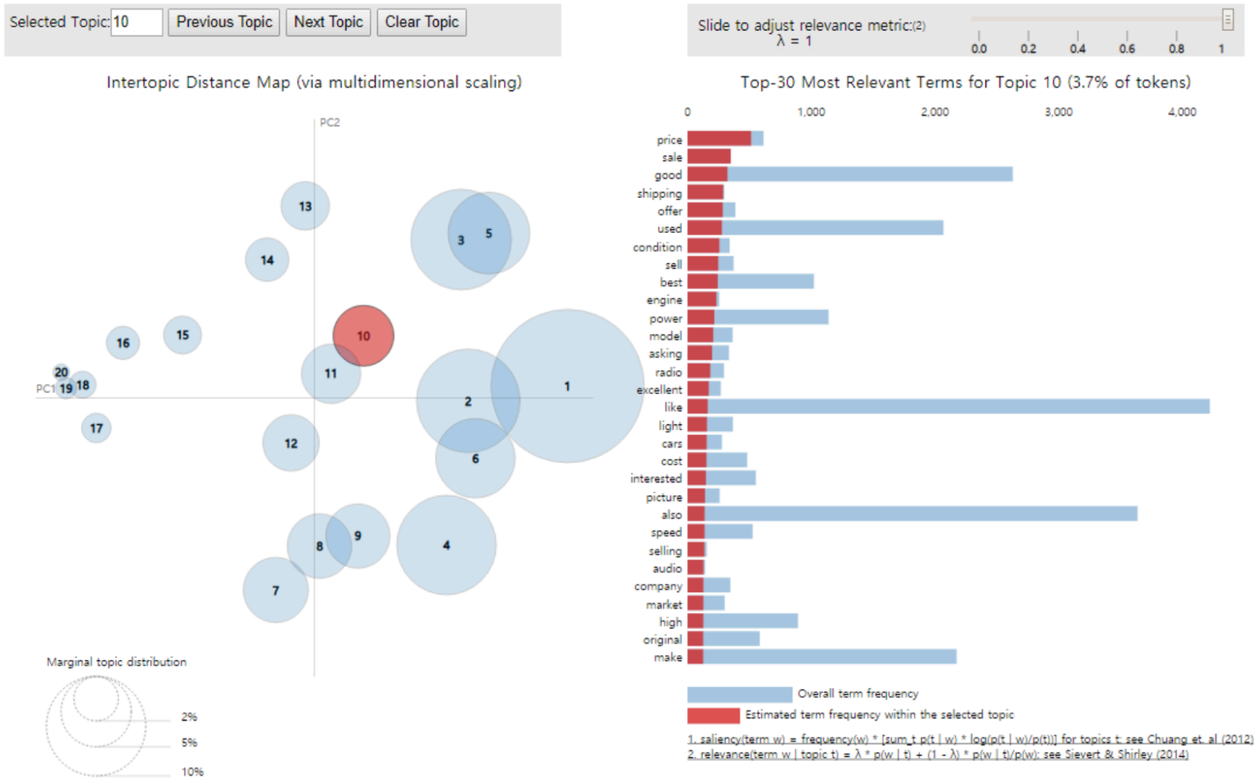
```
[ ]  1 doc_components = lda_model.transform(X[:5])
     2
     3 def get_doc_topic(components, n=5):
     4     for idx, doc in enumerate(components):
     5         print("Doc %d:" % (idx+1), [[f"Topic {i}", doc[i].round(2)] for i in doc.argsort()[:-n - 1:-1]])
     6
     7 get_doc_topic(doc_components, 3)
```

```
Doc 1: [['Topic 1', 0.55], ['Topic 3', 0.05], ['Topic 9', 0.05]]
Doc 2: [['Topic 5', 0.55], ['Topic 3', 0.05], ['Topic 9', 0.05]]
Doc 3: [['Topic 4', 0.63], ['Topic 3', 0.04], ['Topic 9', 0.04]]
Doc 4: [['Topic 6', 0.54], ['Topic 9', 0.24], ['Topic 3', 0.03]]
Doc 5: [['Topic 9', 0.53], ['Topic 6', 0.2], ['Topic 3', 0.03]]
```

LDA 실습 : 뉴스 헤드라인 데이터

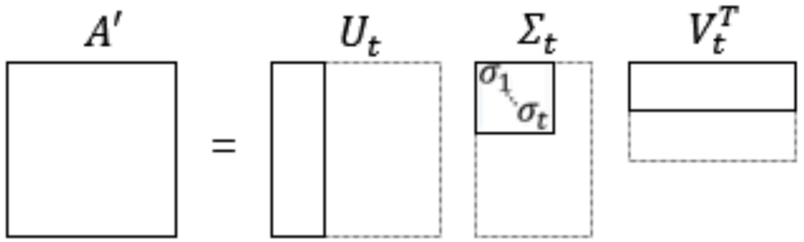
▶ gensim과 pyLDAvis.gensim을 이용하면 손쉽게 시각화도 가능합니다.

```
import pyLDAvis.gensim
pyLDAvis.enable_notebook()
vis = pyLDAvis.gensim.prepare(ldamodel, corpus, dictionary)
pyLDAvis.display(vis)
```



Wrap-Up

LSA



Truncated SVD를 이용하여

DTM, TF-IDF를 저차원의 벡터들로 분해함

U의 행벡터는 문서 표현

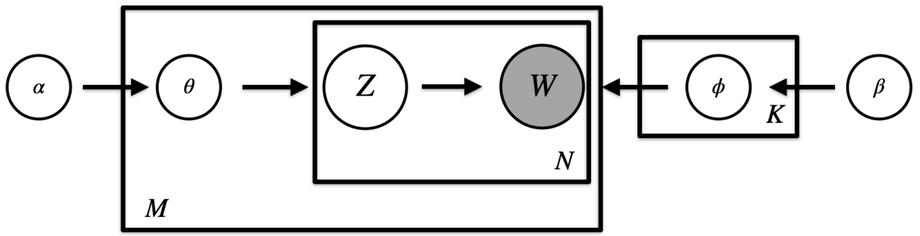
VT의 열벡터는 단어 표현

S의 대각값은 토픽의 중요도

상대적 비교는 가능하나 절대적인 의미는 없음

문서별 단어 빈도를 단순히 분석하기만 했음

LDA



확률 모델을 이용하여

“문서 -> 토픽 -> 단어” 가정

문서별 토픽 확률

토픽별 단어 확률

결과들이 곧바로 확률을 의미

단어의 생성 과정을 reverse-engineering