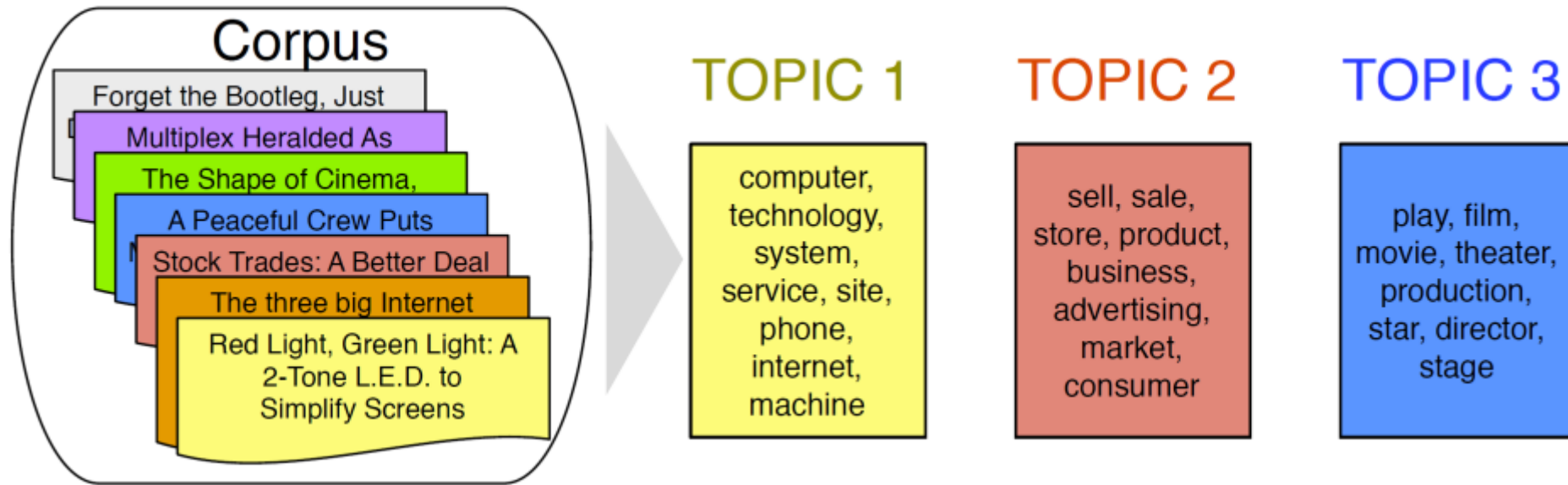


Topic Modeling

LSA , LDA

토픽 모델링 (Topic Modeling)



잠재 의미 분석(Latent Semantic Analysis, LSA)

Bag of words 기반의 DTM(document-term matrix), TF-IDF는 단어의 의미를 고려하지 못한다는 단점이 있음

→ LSA: DTM의 잠재된(Latent) 의미를 이끌어내는 방법

특이값 분해(Singular Value Decomposition, SVD)

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

$U : m \times m$ 직교행렬 ($AA^T = U(\Sigma \Sigma^T)U^T$)

$V : n \times n$ 직교행렬 ($A^T A = V(\Sigma^T \Sigma)V^T$)

$\Sigma : m \times n$ 직사각 대각행렬

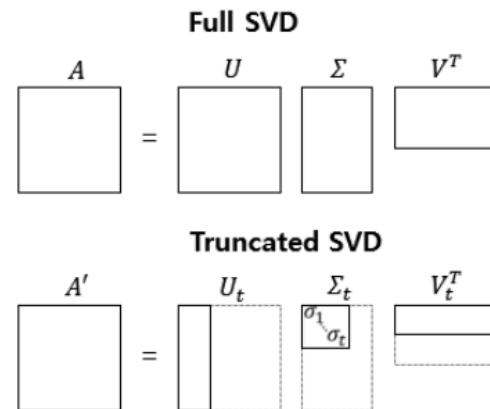
$$\Sigma = \begin{bmatrix} 12.4 & 0 & 0 \\ 0 & 9.5 & 0 \\ 0 & 0 & 1.3 \end{bmatrix}$$

대각의 숫자는 행렬 A의 특잇값(singular value)

차원 축소, 노이즈 제거, 토픽 모델링 등 분야에서 활용

절단된 SVD(Truncated SVD):

- 대각 행렬 Σ 의 대각 원소의 값 중에서 상위값 t 개만 남김, U 행렬과 V 행렬의 t 열까지만 남김
- 계산 비용이 낮아지는 것 외에도 상대적으로 중요하지 않은 정보를 삭제하는 효과
- t 는 토픽의 수



잠재 의미 분석(Latent Semantic Analysis, LSA)

LSA: DTM이나 **TF-IDF 행렬**에 **truncated SVD** 사용해 차원을 축소

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

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

예시 (A는 dtm 행렬)

행렬 U :

$\begin{bmatrix} -0.24 & 0.75 & 0. & -0.62 \\ -0.51 & 0.44 & -0. & 0.74 \\ -0.83 & -0.49 & -0. & -0.27 \\ -0. & -0. & 1. & 0. \end{bmatrix}$

특이값 벡터 :

$[2.69 \ 2.05 \ 1.73 \ 0.77]$

직교행렬 VT :

$\begin{bmatrix} -0. & -0.31 & -0.31 & -0.28 & -0.8 & -0.09 & -0.28 & -0. & -0. \\ 0. & 0.24 & -0.24 & 0.58 & -0.26 & 0.37 & 0.58 & -0. & -0. \\ 0.58 & -0. & 0. & 0. & -0. & 0. & -0. & 0.58 & 0.58 \\ 0. & -0.35 & -0.35 & 0.16 & 0.25 & -0.8 & 0.16 & -0. & -0. \\ -0. & -0.78 & -0.01 & -0.2 & 0.4 & 0.4 & -0.2 & 0. & 0. \\ -0.29 & 0.31 & -0.78 & -0.24 & 0.23 & 0.23 & 0.01 & 0.14 & 0.14 \\ -0.29 & -0.1 & 0.26 & -0.59 & -0.08 & -0.08 & 0.66 & 0.14 & 0.14 \\ -0.5 & -0.06 & 0.15 & 0.24 & -0.05 & -0.05 & -0.19 & 0.75 & -0.25 \\ -0.5 & -0.06 & 0.15 & 0.24 & -0.05 & -0.05 & -0.19 & -0.25 & 0.75 \end{bmatrix}$

- 축소된 U (4×2)는 문서의 개수 \times 토픽의 수 t 의 크기
 U 의 각 행은 잠재 의미를 표현하기 위해 수치화 된 각각의 **문서 벡터**

- 축소된 V^T (2×9)는 토픽의 수 $t \times$ 단어의 개수의 크기
 V^T 의 각 열은 잠재 의미를 표현하기 위해 수치화 된 각각의 **단어 벡터**

이 문서 벡터들과 단어 벡터들을 통해 다른 문서의 유사도, 다른 단어의 유사도, 단어(쿼리)로부터 문서의 유사도를 구하는 것들이 가능

잠재 의미 분석(Latent Semantic Analysis, LSA)

토픽 모델링 예시 축소 차원에서 근접 단어들을 토픽으로 묶음

```
vectorizer = TfidfVectorizer(stop_words='english', max_features= 1000, # 상위 1,000개의 단어들  
                             존  
                             max_df = 0.5, smooth_idf=True)
```

```
X = vectorizer.fit_transform(news_df['clean_doc'])
```

TF-IDF 행렬의 크기 : (11314, 1000)

```
svd_model = TruncatedSVD(n_components=20, algorithm='randomized', n_iter=100, random_state=1234567890)  
svd_model.fit(X)
```

```
np.shape(svd_model.components_) (20, 1000)
```

```
terms = vectorizer.get_feature_names() # 단어 집합. 1,000개의 단어가 저장됨.
```

```
def get_topics(components, feature_names, n=5):  
    for idx, topic in enumerate(components):  
        print("Topic %d:" % (idx+1), [(feature_names[i], topic[i].round(5)) for i in topic.argsort()  
t()[:-n - 1:-1]])  
get_topics(svd_model.components_, terms)
```

잠재 의미 분석(Latent Semantic Analysis, LSA)

토픽 모델링 예시

1. 문서-토픽 행렬 (U):

- 각 행은 원본 문서를 나타내고, 각 열은 하나의 토픽을 나타냅니다.
- 이 행렬의 값은 해당 문서가 각 토픽과 얼마나 관련이 있는지를 나타내는 가중치입니다. 값이 클수록 문서가 그 토픽과 더 관련이 깊다는 것을 의미합니다.

2. 토픽의 중요도를 나타내는 대각 행렬 (Σ):

- 이 대각선에 위치한 값들은 각 토픽의 중요도 또는 "특이값"을 나타냅니다.
- 값이 큰 토픽은 데이터 세트 전체에 걸쳐 더 많은 정보를 담고 있다고 해석할 수 있습니다. 이를 통해 가장 중요한 토픽을 식별할 수 있습니다.

3. 토픽-단어 행렬 (V^T):

- 각 행은 하나의 토픽을 나타내고, 각 열은 원본 데이터 세트의 단어를 나타냅니다.
- 이 행렬의 값은 특정 단어가 토픽에 속하는 정도를 나타내는 가중치입니다. 값이 클수록 해당 단어는 그 토픽을 잘 대표한다는 의미입니다.

잠재 의미 분석(Latent Semantic Analysis, LSA)

토픽 모델링 예시

```
Topic 1: [('like', 0.2138), ('know', 0.20031), ('people', 0.19334), ('think', 0.17802), ('good', 0.15105)]
Topic 2: [('thanks', 0.32918), ('windows', 0.29093), ('card', 0.18016), ('drive', 0.1739), ('mail', 0.15131)]
Topic 3: [('game', 0.37159), ('team', 0.32533), ('year', 0.28205), ('games', 0.25416), ('season', 0.18464)]
Topic 4: [('drive', 0.52823), ('scsi', 0.20043), ('disk', 0.15518), ('hard', 0.15511), ('card', 0.14049)]
Topic 5: [('windows', 0.40544), ('file', 0.25619), ('window', 0.1806), ('files', 0.16196), ('program', 0.14009)]
```

잠재 의미 분석(Latent Semantic Analysis, LSA)

LSA는 쉽고 빠르게 구현이 가능할 뿐만 아니라 단어의 잠재적인 의미를 이끌어낼 수 있어 문서의 유사도 계산 등에서 좋은 성능을 보임

하지만 SVD의 특성상 이미 계산된 LSA에 새로운 데이터를 추가하여 계산하려고 하면 처음부터 다시 계산해야 함
즉, **새로운 정보에 대해 업데이트가 어려움**

→ Word2Vec 등 단어의 의미를 벡터화할 수 있는 또 다른 방법론인 인공 신경망 기반의 방법론 사용

SVD는 normally distribute을 가정하지만, term occurrence는 보통 normally distributed 되지 않음

잠재 디리클레 할당(Latent Dirichlet Allocation, LDA)

잠재 디리클레 할당(Latent Dirichlet Allocation, LDA)은 **토픽 모델링**의 대표적인 알고리즘

LDA는 문서들은 토픽들의 혼합으로 구성되어 있으며, 토픽들은 **확률 분포**에 기반하여 단어들을 생성한다고 가정
LDA는 **각 문서의 토픽 분포**와 **각 토픽 내의 단어 분포**를 추정

데이터가 주어지면, LDA는 문서가 생성되던 과정을 역추적

LDA는 DTM 또는 TF-IDF 행렬을 입력으로 함. 즉, LDA는 단어의 순서는 신경쓰지 않음

문서1 : 저는 사과랑 바나나를 먹어요

문서2 : 우리는 귀여운 강아지가 좋아요

문서3 : 저의 깜찍하고 귀여운 강아지가 바나나를 먹어요

<각 문서의 토픽 분포>

문서1 : 토픽 A 100%

문서2 : 토픽 B 100%

문서3 : 토픽 B 60%, 토픽 A 40%

<각 토픽의 단어 분포>

토픽A : 사과 20%, 바나나 40%, 먹어요 40%, 귀여운 0%, 강아지 0%, 깜찍하고 0%, 좋아요 0%

토픽B : 사과 0%, 바나나 0%, 먹어요 0%, 귀여운 33%, 강아지 33%, 깜찍하고 16%, 좋아요 16%

잠재 디리클레 할당(Latent Dirichlet Allocation, LDA)

LDA 실행 순서

1. 사용자는 **토픽의 개수 k**를 설정

LDA는 토픽의 개수 k를 입력 받으면, k개의 토픽이 M개의 전체 문서에 걸쳐 분포되어 있다고 가정

2. 모든 단어를 k개 중 **하나의 토픽에 할당**

모든 문서의 모든 단어에 대해서 k개 중 하나의 토픽을 랜덤으로 할당

→ 각 문서는 토픽을 가지며, 토픽은 단어 분포를 가지는 상태

3. 이제 모든 문서의 모든 단어에 대해서 아래 사항을 **반복 진행(iterative)**

✓ 어떤 문서의 각 단어 w는 자신은 잘못된 토픽에 할당되어져 있지만, 다른 단어들은 전부 올바른 토픽에 할당되어져 있는 상태라고 가정

doc1					
word	apple	banana	apple	dog	dog
topic	B	B	???	A	A

doc2					
word	cute	book	king	apple	apple
topic	B	B	B	B	B

✓ 이에 따라 단어 w는 아래의 **두 가지 기준**에 따라서 토픽 재할당

- $p(\text{topic } t \mid \text{document } d)$: 문서 d의 단어들 중 토픽 t에 해당하는 단어들의 비율
- $p(\text{word } w \mid \text{topic } t)$: 각 토픽들 t에서 해당 단어 w의 분포

doc1					
word	apple	banana	apple	dog	dog
topic	B	B	???	A	A

doc2					
word	cute	book	king	apple	apple
topic	B	B	B	B	B

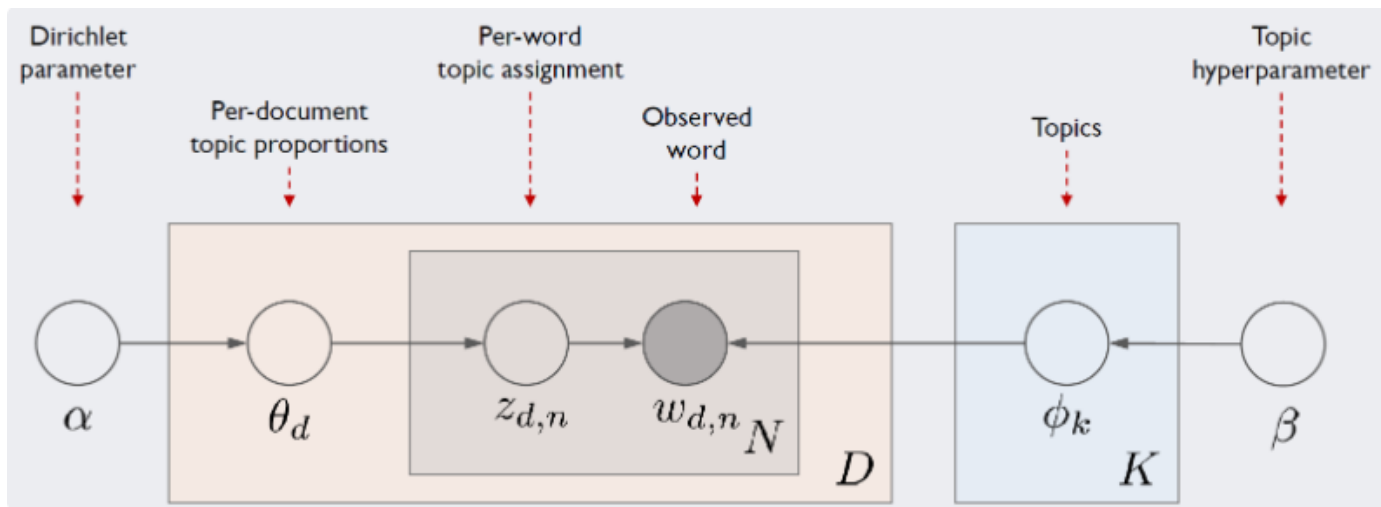
doc1					
word	apple	banana	apple	dog	dog
topic	B	B	???	A	A

doc2					
word	cute	book	king	apple	apple
topic	B	B	B	B	B

잠재 디리클레 할당(Latent Dirichlet Allocation, LDA)

LDA 모델 아키텍처

- LDA의 문서생성과정



- D는 말뭉치 전체 문서 개수, K는 전체 토픽 수, N은 d번째 문서의 단어 수
- 네모 칸은 해당 횟수만큼 반복하라는 의미, 동그라미는 변수
- 화살표가 시작되는 변수는 조건, 화살표가 향하는 변수는 결과에 해당하는 변수

우리가 관찰 가능한 변수는 d번째 문서에 등장한 n번째 단어 $w_{d,n}$ 유일(음영 표시)
이 정보만을 가지고 하이퍼파라미터(사용자 지정) α, β 를 제외한 모든 잠재 변수를 추정

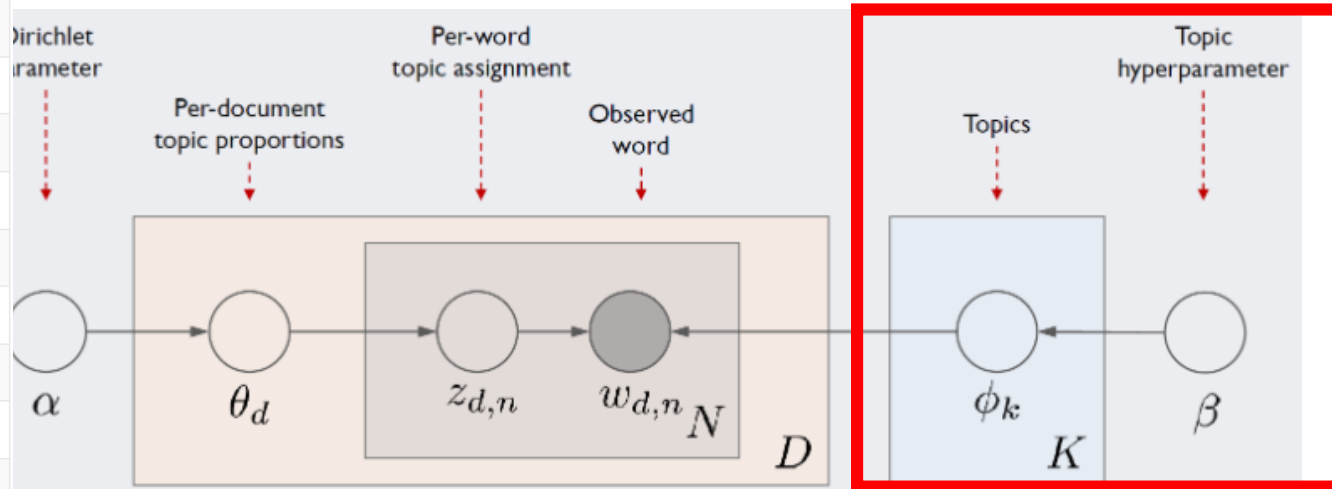
$$p(\phi_{1:K}, \theta_{1:D}, z_{1:D}, w_{1:D}) = \prod_{i=1}^K p(\phi_i | \beta) \prod_{d=1}^D p(\theta_d | \alpha) \left\{ \prod_{n=1}^N p(z_{d,n} | \theta_d) p(w_{d,n} | \phi_{1:K}, z_{d,n}) \right\}$$

잠재 디리클레 할당(Latent Dirichlet Allocation, LDA)

LDA 모델 아키텍처

- ϕ_k 는 k번째 토픽에 해당하는 벡터
 - ϕ_k 의 각 요소값은 해당 단어가 k번째 토픽에서 차지하는 비중
 - ϕ_k 의 각 요소는 확률이므로 모든 요소의 합은 1이 됨. 벡터 길이는 말뭉치 전체 단어 수.
 - ϕ_k 는 하이퍼파라미터 β 에 영향을 받으며, 이는 LDA가 토픽의 단어비중 ϕ_k 이 디리클레분포를 따른다는 가정을 취하기 때문 $\phi_k \sim Dir(\beta)$
 - ✓ 디리클레분포란 k차원의 실수 벡터 중 벡터의 요소가 양수이며 모든 요소를 더한 값이 1인 경우에 확률값이 정의되는 연속확률분포

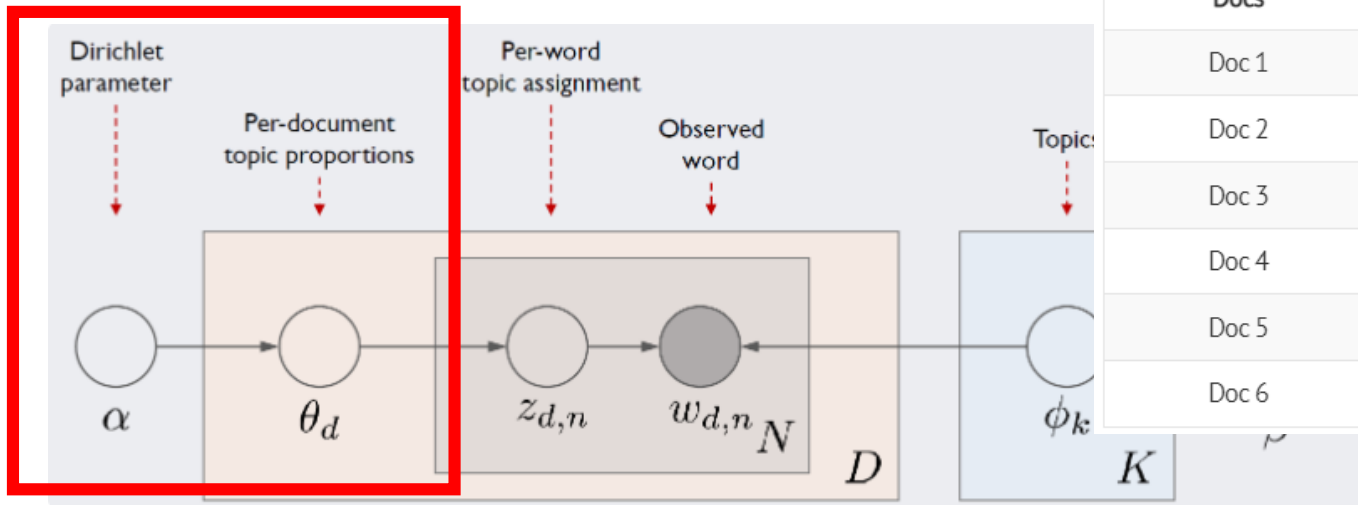
Terms	Topic 1	Topic 2	Topic 3
Baseball	0.000	0.000	0.200
Basketball	0.000	0.000	0.267
Boxing	0.000	0.000	0.133
Money	0.231	0.313	0.400
Interest	0.000	0.312	0.000
Rate	0.000	0.312	0.000
Democrat	0.269	0.000	0.000
Republican	0.115	0.000	0.000
Cocus	0.192	0.000	0.000
President	0.192	0.063	0.000



잠재 디리클레 할당(Latent Dirichlet Allocation, LDA)

LDA 모델 아키텍처

- θ_d 는 d번째 문서가 가진 토픽 비중을 나타내는 벡터
 - θ_d 의 각 요소값은 k번째 토픽이 해당 d번째 문서에서 차지하는 비중
 - θ_d 는 확률이므로 모든 요소의 합은 1. 벡터 길이는 전체 토픽 개수 K.
 - θ_d 는 하이퍼파라미터 α 에 영향을 받으며, 이는 LDA가 문서의 토픽비중 θ_d 이 디리클레분포를 따른다는 가정을 취하기 때문 $\theta_d \sim \text{Dir}(\alpha)$

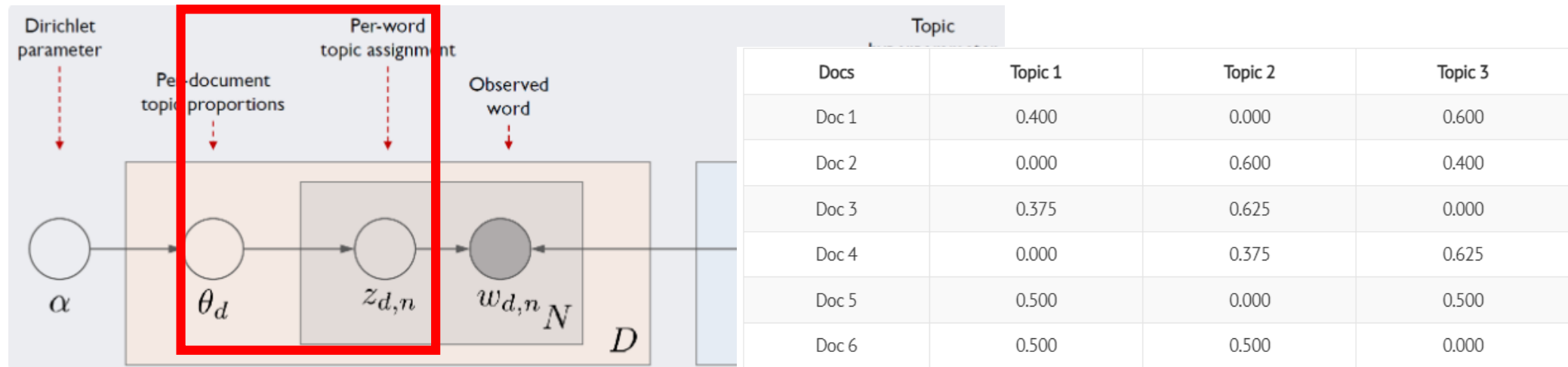


Docs	Topic 1	Topic 2	Topic 3
Doc 1	0.400	0.000	0.600
Doc 2	0.000	0.600	0.400
Doc 3	0.375	0.625	0.000
Doc 4	0.000	0.375	0.625
Doc 5	0.500	0.000	0.500
Doc 6	0.500	0.500	0.000

잠재 디리클레 할당(Latent Dirichlet Allocation, LDA)

LDA 모델 아키텍처

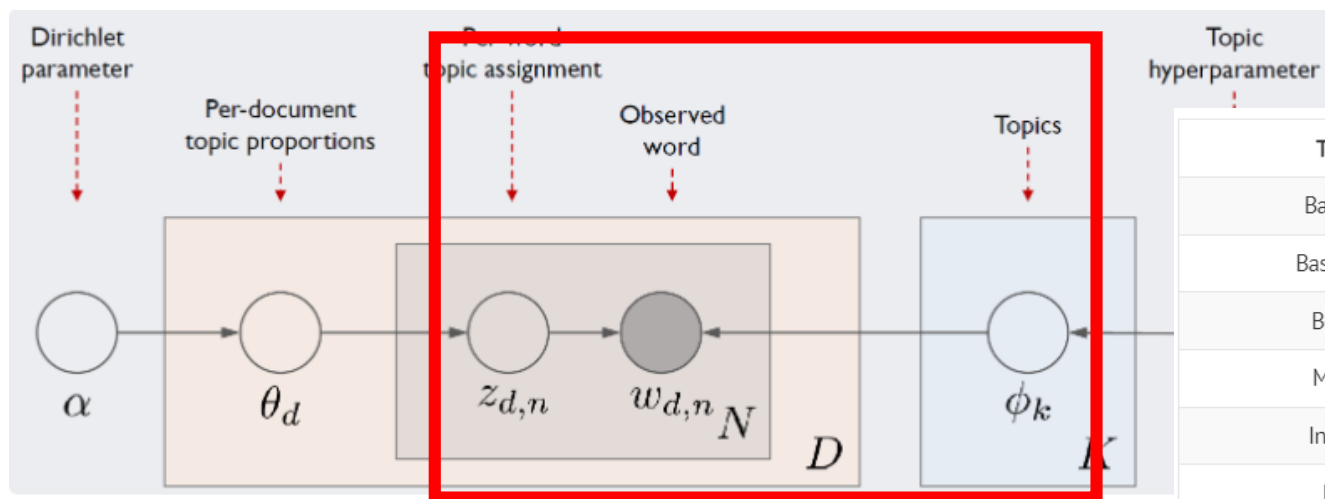
- $z_{d,n}$ 은 d번째 문서 n번째 단어가 어떤 토픽에 해당하는지 할당해주는 역할
 - 예: 세번째 문서의 첫번째 단어 $z_{3,1}$ 는 'Topic2'일 가능성이 높음.



잠재 디리클레 할당(Latent Dirichlet Allocation, LDA)

LDA 모델 아키텍처

- $w_{d,n}$ 은 문서에 등장하는 단어를 할당해주는 역할
 - ϕ_k 와 $z_{d,n}$ 에 동시에 영향을 받음.
 - 직전 예에서 $z_{3,1}$ 이 실제로 Topic2에 할당됐다고 쳤을 때, ϕ_2 를 보면 $w_{3,1}$ 은 Money가 될 가능성이 높음



Terms	Topic 1	Topic 2	Topic 3
Baseball	0.000	0.000	0.200
Basketball	0.000	0.000	0.267
Boxing	0.000	0.000	0.133
Money	0.231	0.313	0.400
Interest	0.000	0.312	0.000
Rate	0.000	0.312	0.000
Democrat	0.269	0.000	0.000
Republican	0.115	0.000	0.000
Cocus	0.192	0.000	0.000
President	0.192	0.063	0.000

잠재 디리클레 할당(Latent Dirichlet Allocation, LDA)

LDA inference

이제 $w_{d,n}$ 를 가지고 잠재변수를 역으로 추정하는 inference 과정을 살펴보자.

LDA가 가정하는 문서생성과정의 합리적이라면 해당 확률과정이 갖고 있는 말뭉치를 제대로 설명할 수 있어야함.
즉, **토픽의 단어분포와 문서의 토픽분포의 결합확률이 커지도록 해야 함.**

$$p(\phi_{1:K}, \theta_{1:D}, z_{1:D}, w_{1:D}) = \prod_{i=1}^K p(\phi_i | \beta) \prod_{d=1}^D p(\theta_d | \alpha) \left\{ \prod_{n=1}^N p(z_{d,n} | \theta_d) p(w_{d,n} | \phi_{1:K}, z_{d,n}) \right\}$$

위 수식에서 사용자가 지정한 α, β 와 관찰가능한 $w_{d,n}$ 을 제외한 모든 변수는 미지수

따라서 사후확률(posterior) $p(\phi, \theta, z | w)$ 를 최대로 만드는 z, ϕ, θ 를 찾아야 함.

$$p(\phi, \theta, \mathbf{z} | \mathbf{w}) = \frac{p(\phi, \theta, \mathbf{z}, \mathbf{w})}{\int_{\phi} \int_{\theta} \sum_{\mathbf{z}} p(\phi, \theta, \mathbf{z}, \mathbf{w})}$$

그러나, 이를 계산하는 것은 매우 어려움 (분모인 $p(w)$ 을 계산할 수 없음)

→ Approximate : Collapsed Gibbs sampling을 사용

잠재 디리클레 할당(Latent Dirichlet Allocation, LDA)

Collapsed Gibbs Sampling

✓ Gibbs Sampling

- Draw a conditioned on b, c
- Draw b conditioned on a, c
- Draw c conditioned on a, b

✓ Collapsed Gibbs Sampling

- Draw a conditioned on c
- Draw c conditioned on a
- b is collapsed out during the sampling process

- Gibbs sampling procedure boils down to estimate

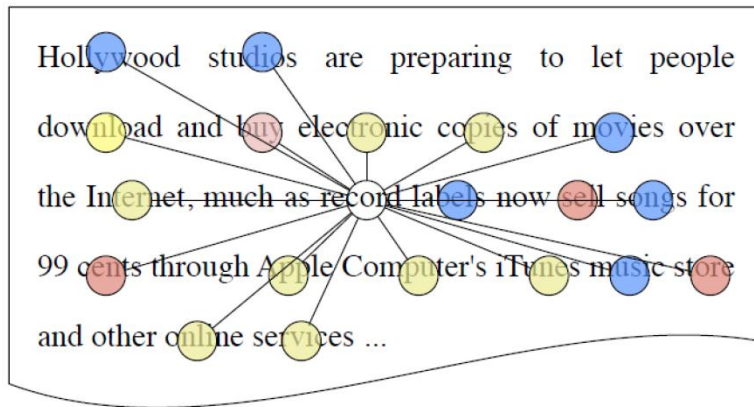
$$p(z_i = j | \mathbf{z}_{-i}, \mathbf{w})$$

- ✓ θ and ϕ are integrated out

computer,
technology,
system,
service, site,
phone,
internet,
machine

sell, sale,
store, product,
business,
advertising,
market,
consumer

play, film,
movie, theater,
production,
star, director,
stage



잠재 디리클레 할당(Latent Dirichlet Allocation, LDA)

Collapsed Gibbs Sampling

$$p(z_{d,i} = j | z_{-i}, w) = \frac{n_{d,k} + \alpha_k}{\sum_{i=1}^K (n_{d,i} + \alpha_i)} \times \frac{v_{k,w_{d,n}} + \beta_{w_{d,n}}}{\sum_{j=1}^V (v_{k,j} + \beta_j)} = AB$$

표기	내용
$n_{d,k}$	k 번째 토픽에 할당된 d 번째 문서의 단어 빈도
$v_{k,w_{d,n}}$	전체 말뭉치에서 k 번째 토픽에 할당된 단어 $w_{d,n}$ 의 빈도
$w_{d,n}$	d 번째 문서에 n 번째로 등장한 단어
α	문서의 토픽 분포 생성을 위한 디리클레 분포 파라미터
β	토픽의 단어 분포 생성을 위한 디리클레 분포 파라미터
K	사용자가 지정하는 토픽 수
V	말뭉치에 등장하는 전체 단어 수
A	d 번째 문서가 k 번째 토픽과 맺고 있는 연관성 정도
B	d 번째 문서의 n 번째 단어($w_{d,n}$)가 k 번째 토픽과 맺고 있는 연관성 정도

잠재 디리클레 할당(Latent Dirichlet Allocation, LDA)

디리클레 분포 (Dirichlet distribution)

- Dirichlet distribution

$$p(P = \{p_i\} | \alpha_i) = \frac{\Gamma(\sum_i \alpha_i)}{\prod_i \Gamma(\alpha_i)} \prod_i p_i^{\alpha_i - 1}$$

✓ $\sum_i p_i = 1, p_i \geq 0$

✓ Two parameters

- the scale (or concentration): $\sigma = \sum_i \alpha_i$
- the base measure: $(\alpha'_1, \dots, \alpha'_k), \alpha'_i = \alpha_i / \sigma$

✓ A generalization of Beta

- Beta is a distribution over binomials (in an interval $p \in [0, 1]$)
- Dirichlet is a distribution over Multinomials (in the so-called simplex $\sum_i p_i = 1, p_i \geq 0$)

✓ Dirichlet is the conjugate prior of multinomial

✓ Posterior is also Dirichlet

$$p(P = \{p_i\} | \alpha_i) = \frac{\Gamma(\sum_i \alpha_i)}{\prod_i \Gamma(\alpha_i)} \prod_i p_i^{\alpha_i - 1}$$

$$p(x_1, \dots, x_k | n, p_1, \dots, p_k) = \frac{n!}{\prod_i^k x_i!} p_i^{x_i}$$

$$p(\{p_i\} | x_1, \dots, x_k) = \frac{\Gamma(N + \sum_i \alpha_i)}{\prod_i^k \Gamma(\alpha_i + x_i)} \prod_i p_i^{\alpha_i + x_i - 1}$$

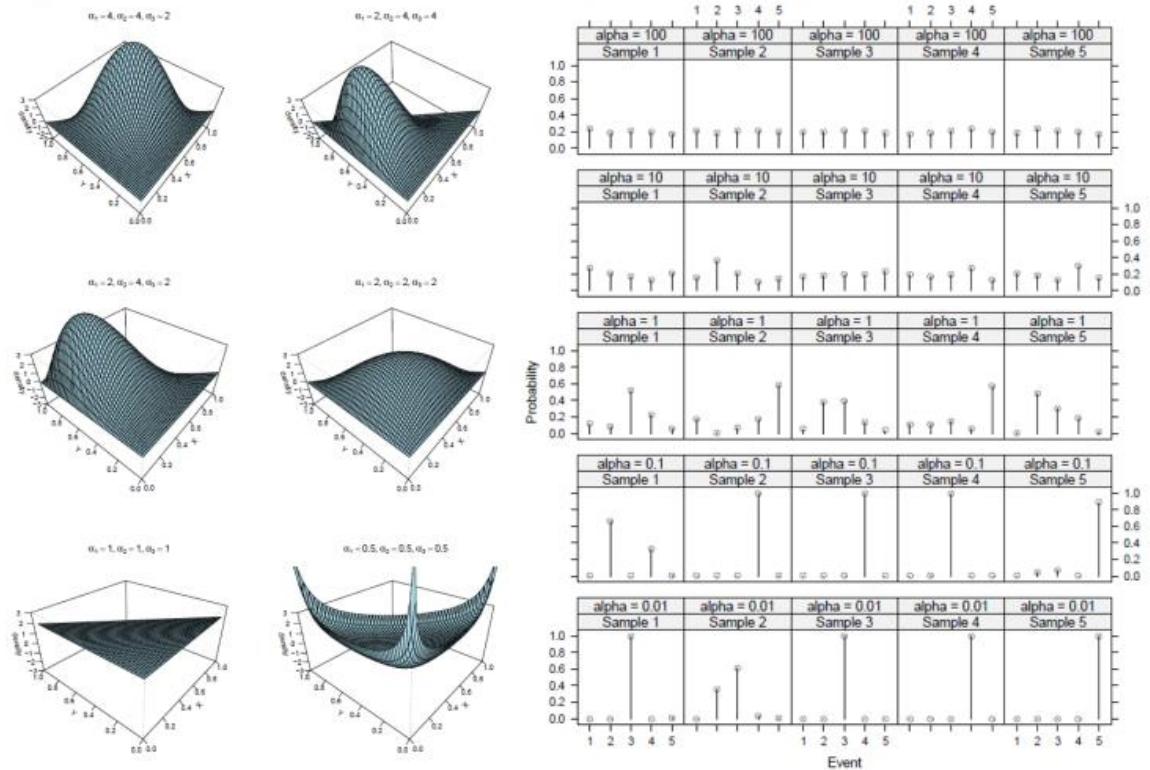
잠재 디리클레 할당(Latent Dirichlet Allocation, LDA)

디리클레 분포 (Dirichlet distribution)

$$p(z_{d,i} = j | z_{-i}, w) = \frac{n_{d,k} + \alpha_j}{\sum_{i=1}^K (n_{d,i} + \alpha_i)} \times \frac{v_{k,w_{d,n}} + \beta_{w_{d,n}}}{\sum_{j=1}^V (v_{k,j} + \beta_j)} = AB$$

- 사용자가 지정하는 하이퍼파라미터 α 존재 덕분에 A가 아예 0으로 되는 일을 막을 수 있게 됨
- 일종의 smoothing 역할
- α 가 클수록 토픽들의 분포가 비슷해지고, 작을수록 특정 토픽이 크게 나타남.
- β 가 B에서 차지하는 역할도 같음

✓ The parameter α controls the mean shape and sparsity of θ



잠재 디리클레 할당(Latent Dirichlet Allocation, LDA)

```
import gensim
NUM_TOPICS = 20 # 20개의 토픽, k=20
ldamodel = gensim.models.ldamodel.LdaModel(corpus, num_topics = NUM_TOPICS, id2word=dictionary, passes=15)
topics = ldamodel.print_topics(num_words=4)
for topic in topics:
    print(topic)
```

Copy

passes는 알고리즘의 동작 횟수

```
(0, '0.015*"drive" + 0.014*"thanks" + 0.012*"card" + 0.012*"system"')
(1, '0.009*"back" + 0.009*"like" + 0.009*"time" + 0.008*"went"')
(2, '0.012*"colorado" + 0.010*"david" + 0.006*"decenso" + 0.005*"tyre"')
(3, '0.020*"number" + 0.018*"wire" + 0.013*"bits" + 0.013*"filename"')
(4, '0.038*"space" + 0.013*"nasa" + 0.011*"research" + 0.010*"medical"')
(5, '0.014*"price" + 0.010*"sale" + 0.009*"good" + 0.008*"shipping"')
(6, '0.012*"available" + 0.009*"file" + 0.009*"information" + 0.008*"version"')
(7, '0.021*"would" + 0.013*"think" + 0.012*"people" + 0.011*"like"')
(8, '0.035*"window" + 0.021*"displav" + 0.017*"widget" + 0.013*"application"')
```

잠재 디리클레 할당(Latent Dirichlet Allocation, LDA)

문서 번호	가장 비중이 높은 토픽	가장 높은 토픽의 비중	각 토픽의 비중
0	0	9.0	0.5071 [(7, 0.30498007), (9, 0.5071116), (11, 0.13196...
1	1	7.0	0.7634 [(7, 0.76344866), (13, 0.02931299), (14, 0.128...
2	2	7.0	0.5224 [(7, 0.522409), (9, 0.36602637), (16, 0.097610...
3	3	10.0	0.5854 [(1, 0.16931751), (5, 0.04911898), (6, 0.04034...
4	4	7.0	0.4215 [(7, 0.42152897), (12, 0.21915697), (17, 0.327...
5	5	7.0	0.3865 [(7, 0.38652688), (8, 0.1446223), (9, 0.244169...
6	6	7.0	0.3182 [(0, 0.30865452), (5, 0.2190474), (6, 0.059171...
7	7	1.0	0.3606 [(1, 0.36058533), (7, 0.34877154), (9, 0.16878...
8	8	14.0	0.5236 [(0, 0.21124822), (7, 0.18776271), (12, 0.0538...
9	9	7.0	0.4333 [(0, 0.0694689), (5, 0.37277842), (7, 0.433252...

LSA : DTM을 차원 축소하여 축소 차원에서 근접 단어들을 토픽으로 묶는다.

LDA : 단어가 특정 토픽에 존재할 확률과 문서에 특정 토픽이 존재할 확률을 결합확률로 추정하여 토픽을 추출한다.

참고자료

- 딥 러닝을 이용한 자연어 처리 입문(<https://wikidocs.net/book/2155>)
- ratsgo's blog (<https://ratsgo.github.io/from%20frequency%20to%20semantics/2017/06/01/LDA/>)
- Unstructured Data Analysis (Graduate) @Korea University (<https://github.com/pilsung-kang/text-analytics>)