

Probabilistic Topic Models

Jeonghun Yoon

Natural Language Processing

Natural language(자연어) : 일상 생활에서 사용하는 언어

Natural language processing(자연어 처리) : 자연어의 의미를 분석하여 컴퓨터가 여러가지 일들을(tasks) 처리할 수 있도록 하는 것

Easy

- Spell checking, Keyword search, Finding synonyms

Medium

- Parsing information from websites, documents, etc

Hard

- Machine translation
- ***Semantic analysis***
- ***Coherence***
- Question answering

Semantic analysis

언어학에서의 의미 분석

- 자연어를 이해하는 기법 중 하나로, 문장의 의미(meaning, semantic)에 근거하여 문장을 해석하는 것을 의미
- Syntax analysis의 반대(lexicon, syntax analysis)

머신러닝에서의 의미 분석

- Corpus에 있는 많은 documents 집합에 내제되어 있는(latent) meanings, concepts, subjects, topics 등을 추정할 수 있는 구조를 생성하는 것을 의미
- 대표적인 의미 분석 기법
 - Latent Semantic Analysis(LSA or LSI)
 - PLSI
 - Latent Dirichlet Allocation(LDA)
 - Hierarchical Dirichlet Processing(HDP)

Semantic analysis

Representation of documents



Axes of a spatial

- Euclidean space에서 정의 가능
- Hard to interpret

Probabilistic topics

- 단어상에 정의된 probability distribution
- Interpretable

1. Axes of a spatial

- LSA

2. Probabilistic topics

- LDA

3. Bayesian Nonparametric

- HDP

LSA (Latent Semantic Analysis)

- LSA(LSI)는 document data의 숨겨진 의미(hidden concept)를 찾아내는 기법이다.
- LSA는 각각의 문서(document)와 단어(word)를 벡터로 표현한다. 벡터내에서 각각의 element는 숨겨진 의미가 될 것이다.

LSA (Latent Semantic Analysis)

d_1 : Romeo and Juliet.

d_2 : Juliet: O happy dagger!

d_3 : Romeo died by dagger.

d_4 : "Live free or die", that's the motto of New-Hampshire

d_5 : Did you know, New-Hampshire is in New-England

Query : dies and dagger

- 3번 문서는 쿼리에 대해서 1등이 될 것이다.
 - 2번, 4번 문서는 그 다음이 될 것이다.
 - 1번, 5번 문서는?
 - ✓ 사람들이 인식하기로는 문서 1번이 문서 5번 보다 주어진 쿼리에 더 맞는 문서이다.
- 컴퓨터도 이러한 추론 과정을 할 수 있을까? 즉 숨겨진 의미를 찾을 수 있을까?

LSA (Latent Semantic Analysis)

matrix A :

live die free new-hampshire

d_1 [[1 1 0 0 0 0 0 0]

d_2 [0 1 1 1 0 0 0 0]

d_3 [1 0 0 1 0 1 0 0]

d_4 [0 0 0 0 1 1 1 1]

d_5 [0 0 0 0 0 0 0 1]]

romeo juliet happy dagger

LSA (Latent Semantic Analysis)

matrix A :

		live	die	free	new-hampshire			
d_1	[1	1	0	0	0	0	0]
d_2	[0	1	1	1	0	0	0]
d_3	[1	0	0	1	0	1	0]
d_4	[0	0	0	0	1	1	1]
d_5	[0	0	0	0	0	0	1]]]
		romeo	juliet	happy	dagger			

doc-doc matrix

matrix $B = AA^T$ doc-doc matrix

문서 i 와 문서 j 가 b 개 의 공통
단어를 가지고 있으면 $B[i,j] = b$

1번 문서에는 romeo, juliet, 2번 문서에는 juliet, happy, dagger

즉 겹쳐지는 것이 1개이므로 $B[1,2] = B[2,1] = 1$

matrix A :

1	1	0	0	0	0	0	0
0	1	1	1	0	0	0	0
1	0	0	1	0	1	0	0
0	0	0	0	1	1	1	1
0	0	0	0	0	0	0	1

5×8

matrix A^T :

1	0	1	0	0
1	1	0	0	0
0	1	0	0	0
0	1	1	0	0
0	0	0	1	0
0	0	1	1	0
0	0	0	1	0
0	0	0	1	1

8×5

matrix $AA^T(B)$:

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

5×5

word-word matrix

matrix $C = A^T A$ word-word matrix

즉, 단어 i 와 단어 j 가 c 개의 문서에서
함께 발생했으면 $C[i, j] = c$

juliet은 1번, 2번 문서에서 나오고, dagger는 2, 3번 문서에서 나온다.

즉 겹쳐지는 것이 1개이므로 $C[2, 4] = B[4, 2] = 1$

matrix A^T :

[[1	0	1	0	0]
[1	1	0	0	0]	
[0	1	0	0	0]	
[0	1	1	0	0]	
[0	0	0	1	0]	
[0	0	1	1	0]	
[0	0	0	1	0]	
[0	0	0	1	1]]	

8×5

matrix A :

[[1	1	0	0	0	0	0]
[0	1	1	1	0	0	0	0]
[1	0	0	1	0	1	0	0]
[0	0	0	0	1	1	1	1]
[0	0	0	0	0	0	0	1]]

5×8

matrix $A^T A(C)$:

[[2	1	0	1	0	1	0	0]
[1	2	1	1	0	0	0	0]	
[0	1	1	1	0	0	0	0]	
[1	1	1	2	0	1	0	0]	
[0	0	0	0	1	1	1	1]	
[1	0	0	1	1	2	1	1]	
[0	0	0	0	1	1	1	1]	
[0	0	0	0	1	1	1	2]]	

8×8

LSA (Latent Semantic Analysis)

SVD 사용!

$A = U\Sigma V^T$, U 는 B 의 eigenvectors이고, V 는 C 의 eigenvectors이다.

$$\Sigma = \begin{bmatrix} 2.285 & 0 & 0 & 0 & 0 \\ 0 & 2.010 & 0 & 0 & 0 \\ 0 & 0 & 1.361 & 0 & 0 \\ 0 & 0 & 0 & 1.118 & 0 \\ 0 & 0 & 0 & 0 & 0.797 \end{bmatrix}$$



singular value

LSA (Latent Semantic Analysis)

Reduced SVD 사용!

$A_k = S_k \Sigma_k U_k^T$, 모든 singular value를 사용할 수 없고, 작은 것들은 제외한다.

k 개의 특이값만 남기는 것이다. 즉 k 개의 "hidden concepts"만 남긴다.

LSA (Latent Semantic Analysis)

$$\Sigma_2 = \begin{bmatrix} 2.285 & 0 \\ 0 & 2.010 \end{bmatrix}$$

$$V_2^T = \text{array}([[0.39615277, 0.28005737], [0.31426806, 0.44953214], [0.17823952, 0.26899154], [0.43836375, 0.36850831], [0.26388058, -0.34592143], [0.52400482, -0.24640466], [0.26388058, -0.34592143], [0.32637322, -0.45966878]])$$

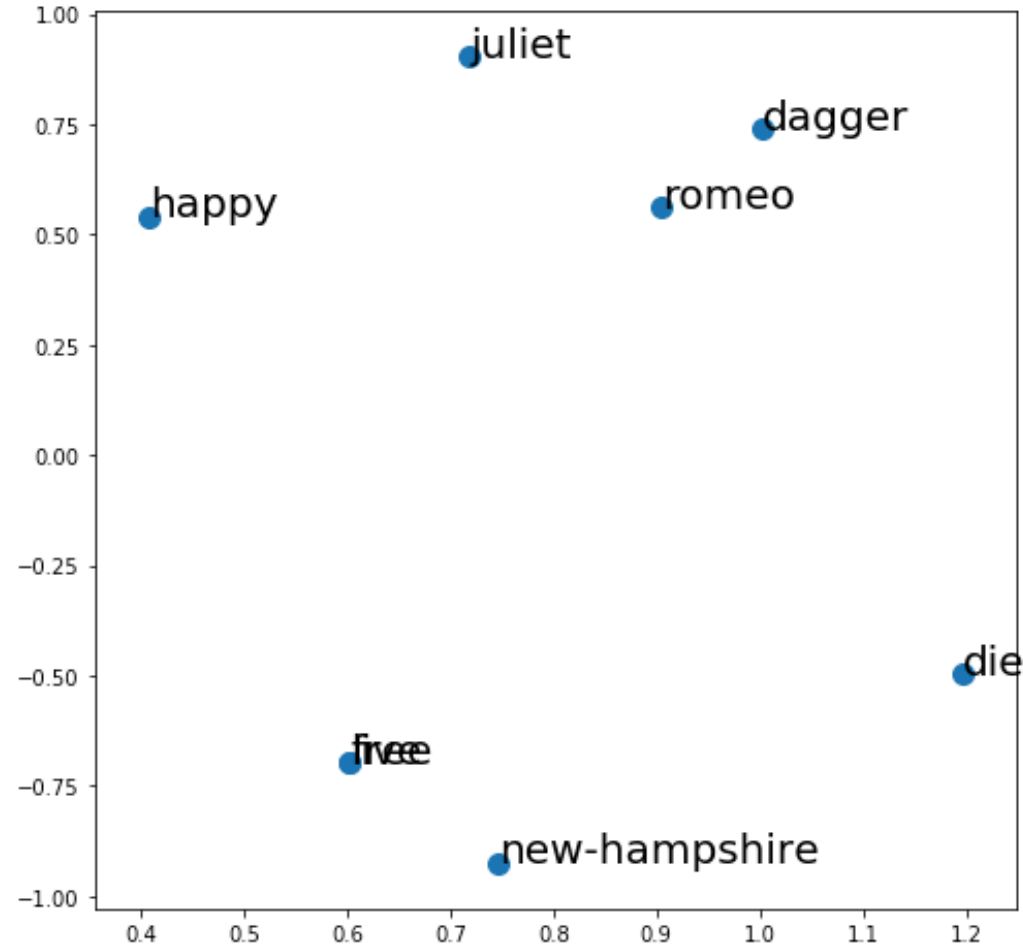
*romeo
juliet
happy
dagger
live
die
free
new-hampshire*

Word vector

$$\Sigma_2 V_2^T = \text{array}([[0.90532712, 0.56298763], [0.71819615, 0.90367568], [0.40733041, 0.54074246], [1.00179178, 0.74079687], [0.60304575, -0.6953914], [1.19750713, -0.49533699], [0.60304575, -0.6953914], [0.74586005, -0.92405295]])$$

*romeo
juliet
happy
dagger
live
die
free
new-hampshire*

LSA (Latent Semantic Analysis)



Word vector의 scatter

LSA (Latent Semantic Analysis)

$$\Sigma_2 = \begin{bmatrix} 2.285 & 0 \\ 0 & 2.010 \end{bmatrix}$$

$$U_2 = \text{array}(\begin{bmatrix} 0.31086574, & 0.36293322, \\ 0.40733041, & 0.54074246, \\ 0.59446137, & 0.20005441, \\ 0.60304575, & -0.6953914, \\ 0.1428143, & -0.22866156, \end{bmatrix}$$

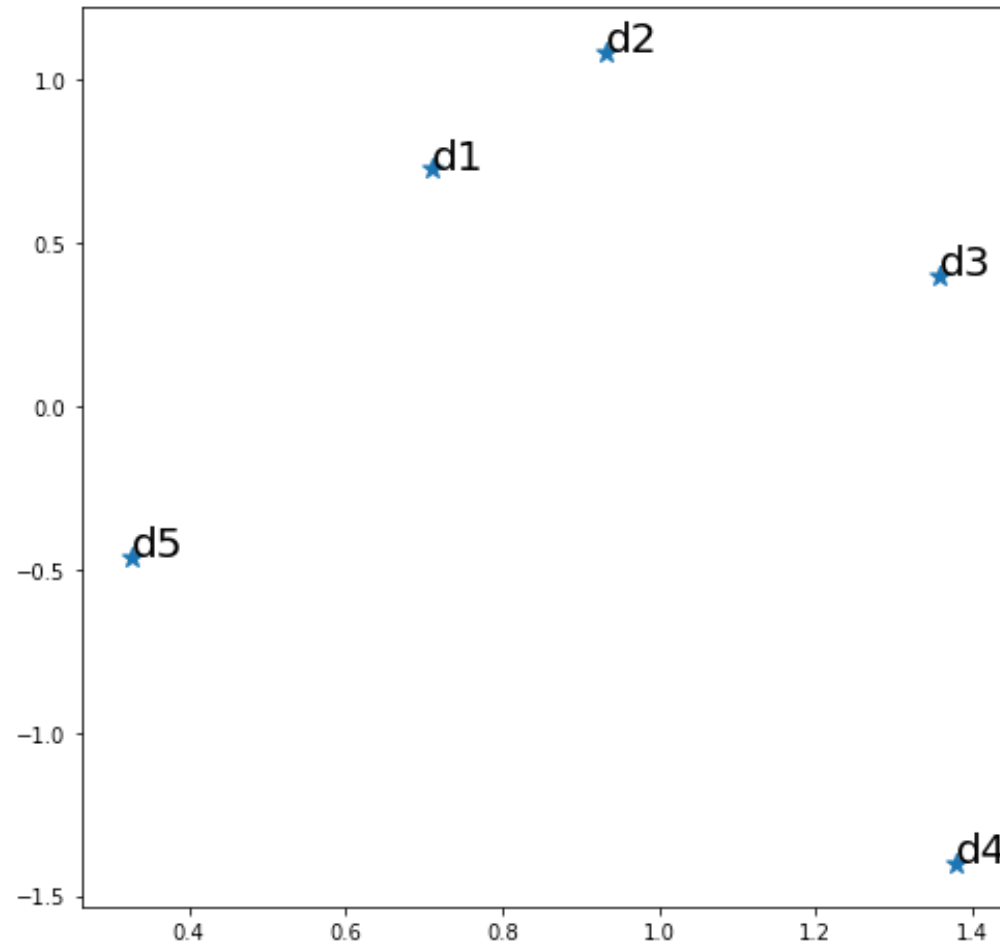
d_1
 d_2
 d_3
 d_4
 d_5

Document vector

$$U_2 \Sigma_2 = \text{array}(\begin{bmatrix} 0.71042084, & 0.7295895, \\ 0.93087134, & 1.08703198, \\ 1.35852135, & 0.40216102, \\ 1.37813921, & -1.39791629, \\ 0.32637322, & -0.45966878, \end{bmatrix})$$

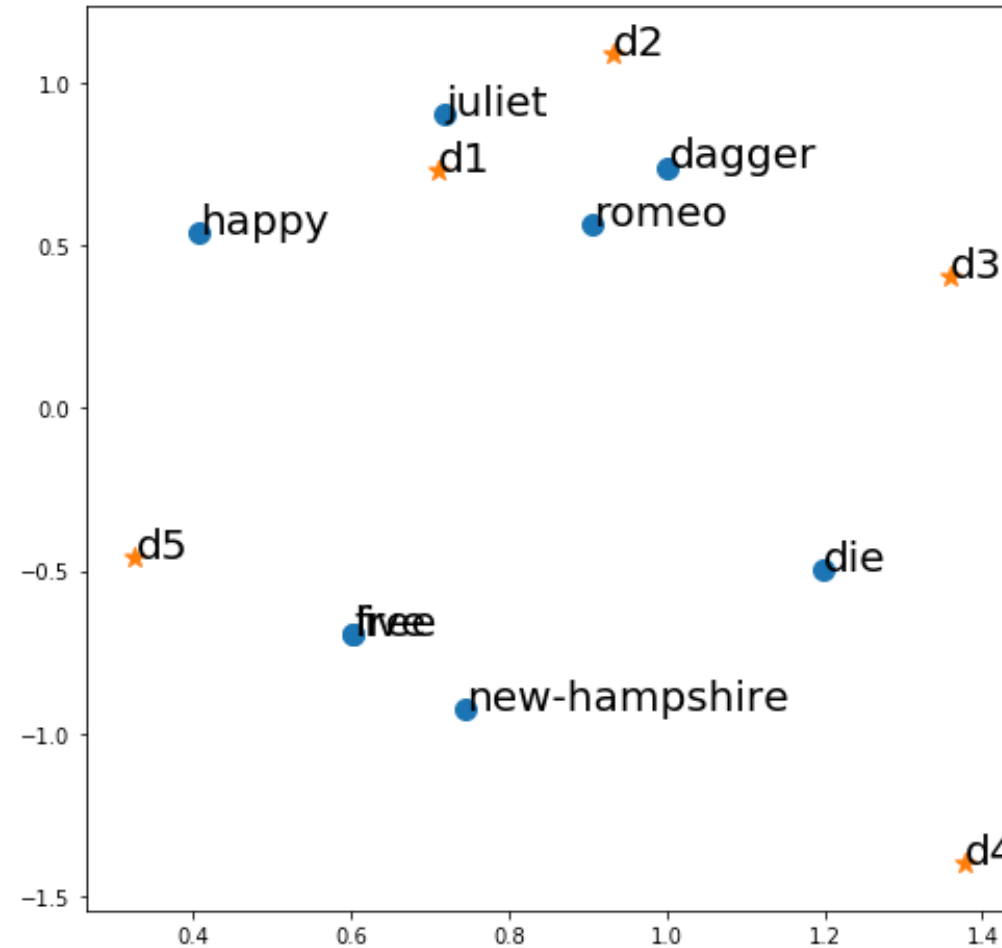
d_1
 d_2
 d_3
 d_4
 d_5

LSA (Latent Semantic Analysis)



Document vector의 scatter

LSA (Latent Semantic Analysis)



Word / Document vector scatter

LSA (Latent Semantic Analysis)

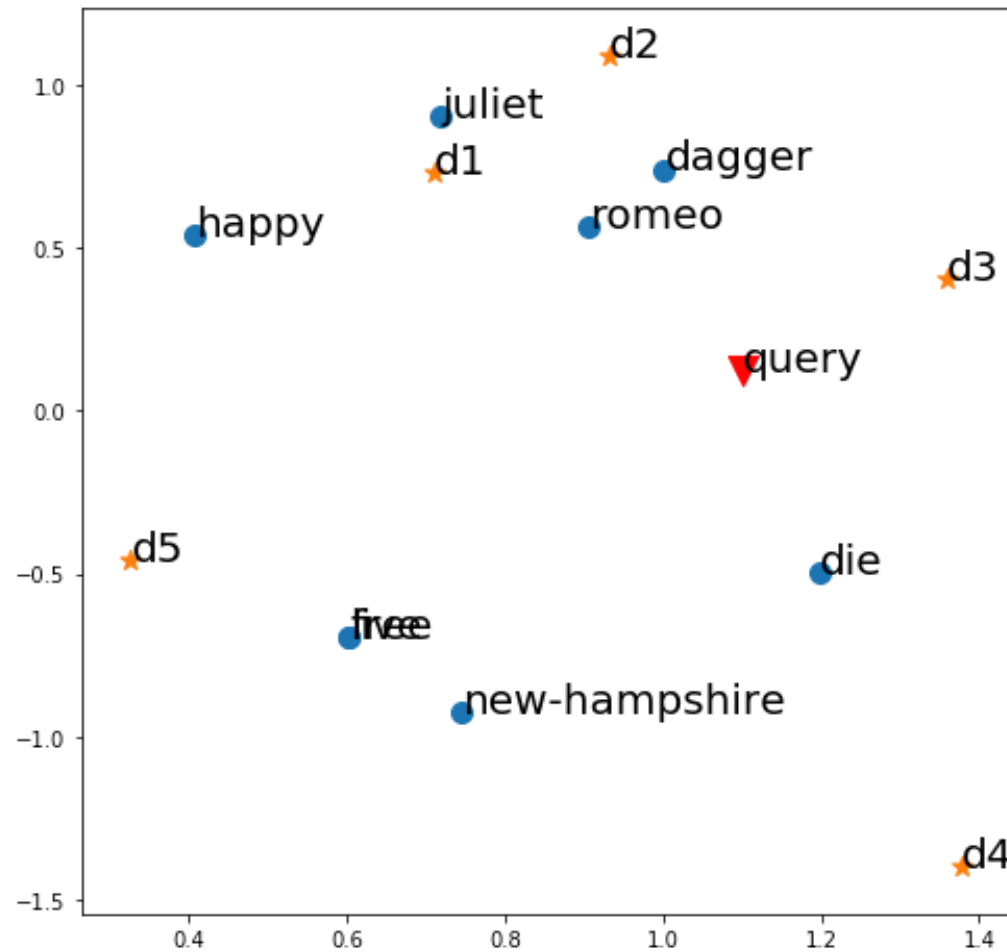
$$q = \frac{q_1 + q_2}{2}$$

$$\text{cosine similarity} = \frac{d_i \cdot q}{|d_i||q|}$$

query : dagger, die

```
result : [(0.9844359912676067, 'Romeo die by dagger.'),  
          (0.7727964887537556, 'Romeo and Juliet.'),  
          (0.7306768205359726, 'Juliet: O happy dagger!'),  
          (0.6187306127613211, "'Live free or die', that's the motto of New-Hampshire"),  
          (0.4849183185073821, 'Did you know, New-Hampshire is in New-England')]
```

LSA (Latent Semantic Analysis)



Word / Document / Query vector의 scatter

1. Axes of a spatial

- LSA

2. Probabilistic topics

- LDA

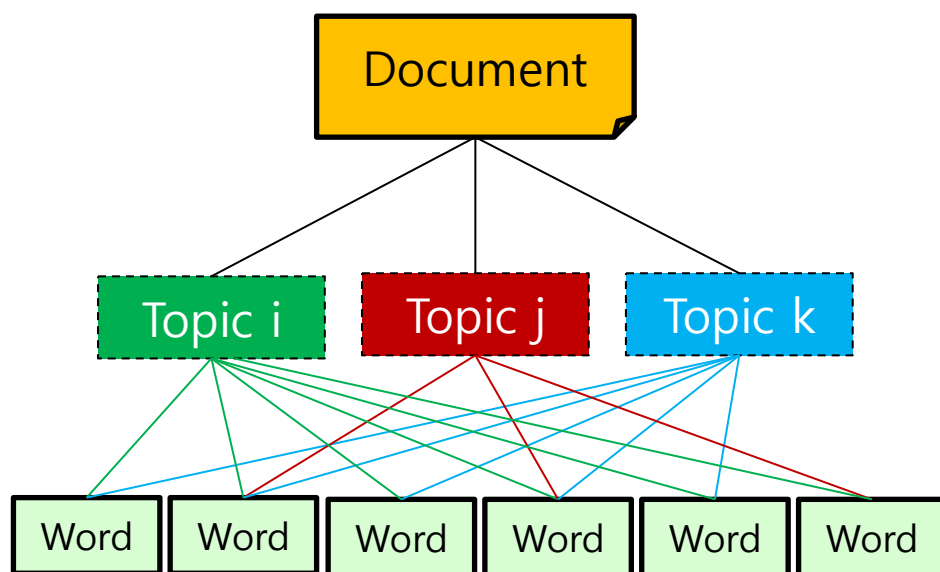
3. Bayesian Nonparametric

- HDP

Topic models

Topic models의 기본 아이디어

- 문서는 토픽들의 혼합 모델이며 각 토픽은 단어상에 정의된 확률분포



Topic 247

word	prob.
DRUGS	.069
DRUG	.060
MEDICINE	.027
EFFECTS	.026
BODY	.023
MEDICINES	.019
PAIN	.016
PERSON	.016
MARIJUANA	.014
LABEL	.012
ALCOHOL	.012
DANGEROUS	.011
ABUSE	.009
EFFECT	.009
KNOWN	.008
PILLS	.008

Topic 5

word	prob.
RED	.202
BLUE	.099
GREEN	.096
YELLOW	.073
WHITE	.048
COLOR	.048
BRIGHT	.030
COLORS	.029
ORANGE	.027
BROWN	.027
PINK	.017
LOOK	.017
BLACK	.016
PURPLE	.015
CROSS	.011
COLORED	.009

Topic 43

word	prob.
MIND	.081
THOUGHT	.066
REMEMBER	.064
MEMORY	.037
THINKING	.030
PROFESSOR	.028
FELT	.025
REMEMBERED	.022
THOUGHTS	.020
FORGOTTEN	.020
MOMENT	.020
THINK	.019
THING	.016
WONDER	.014
FORGET	.012
RECALL	.012

Topic models

예제)

Doc 1 : I like to eat broccoli and bananas.

Doc 2 : I ate a banana and tomato smoothie for breakfast.

Doc 3 : Dogs and cats are cute.

Doc 4 : My sister adopted a cats yesterday.

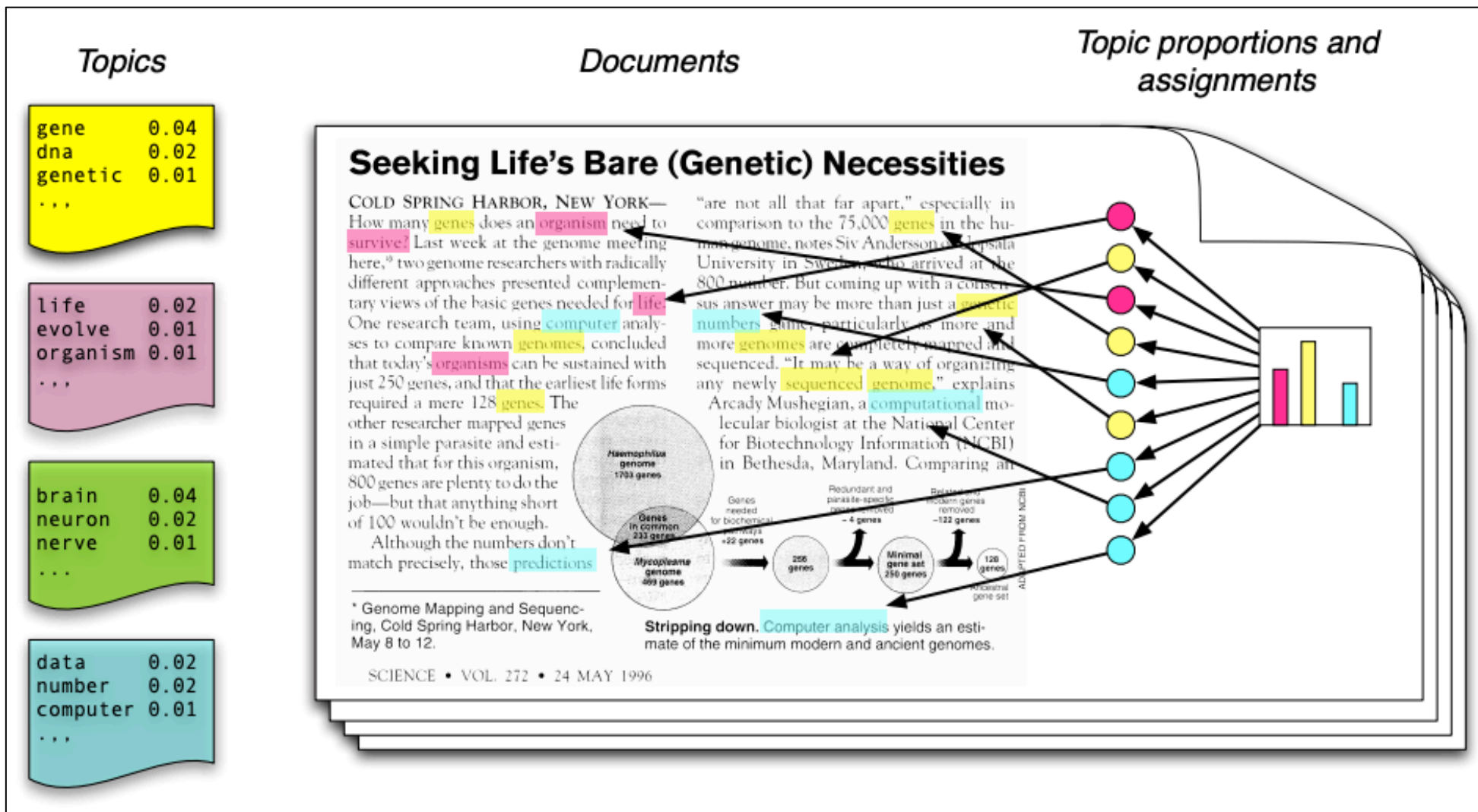
Doc 5 : Look at this cute hamster munching on a piece of broccoli.



- Doc 1 and 2 : 100% topic A
- Doc 3 and 4 : 100% topic B
- Doc 5 : 60% topic A, 40% topic B

- Topic A: 30% broccoli, 15% bananas, 10% breakfast, 10% munching, ...
- Topic B: 20% cats, 20% cute, 15% dogs, 15% hamster, ...

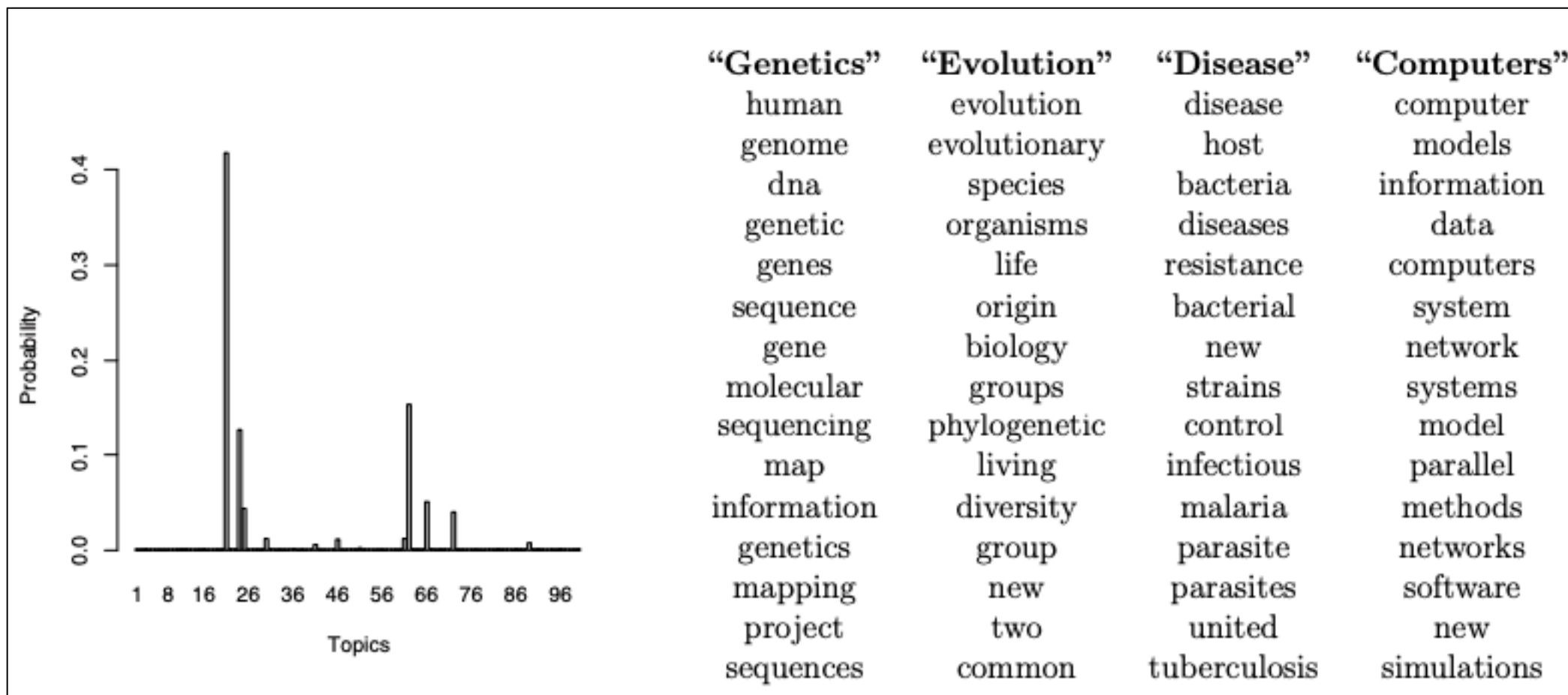
Topic models



Topic models

(Left) 문서에서의 topic proportion

(Right) 문서에서 비중이 높았던 토픽에 대하여,
토픽별 문서내 빈도수가 가장 높은 단어



Probabilistic Topic Models의 구조

모델의 정의에 앞서, 필요한 단어들의 수학적 표기

- Word : $\{1, \dots, V\}$ 를 인덱스로 가지는 vocabulary 상의 items
- Document : N word의 sequence
 - $\mathbb{w} = (w_1, w_2, \dots, w_N)$, w_n : word의 sequence내에서 n 번째에 있는 word
- Corpus : D documents의 collection
 - $\mathcal{C} = \{\mathbb{w}_1, \mathbb{w}_2, \dots, \mathbb{w}_D\}$

Probabilistic Topic Models의 구조

문서 d 의 단어 w_i 대한 분포 :

$$P(w_i) = \sum_{k=1}^K P(w_i|z_i = k)P(z_i = k)$$

- $P(w_i|z_i = k)$: 토픽 k 에서, 단어 w_i 의 probability
 - 각 토픽에서 어떤 단어들이 중요할까?
- $P(z_i = k)$: i 번째 단어에 토픽 k 가 할당되는 probability (즉, 토픽 j 가 i 번째 단어를 위해 샘플링 될 확률)

$\beta_k = P(w|z = k)$: 토픽 k 에서, 단어들의 multinomial distribution

$\theta_d = P(z)$: 문서 d 에서, 토픽들의 multinomial distribution

Latent Dirichlet Allocation의 등장

문서 d 의 단어 w_i 대한 분포 :

$$P(w_i) = \sum_{k=1}^K P(w_i | z_i = k) P(z_i = k)$$

LDA는 Dirichlet distribution을 θ 의 prior로 사용
(Blei et. Al, 2003)

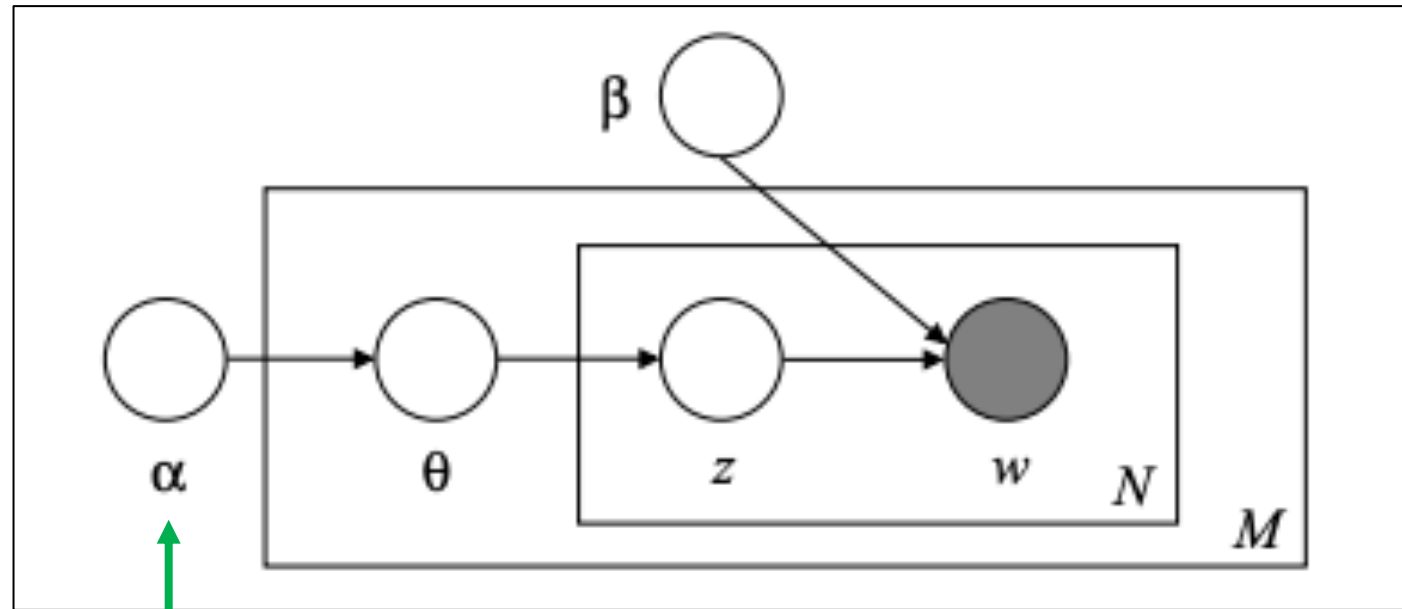
디리클레 분포(Dirichlet distribution)은 multinomial distribution의 켤레 사전 분포로(conjugate prior) 사용

다항 분포(Multinomial distribution) $p = (p_1, \dots, p_K)$ 에 대한 Dirichlet distribution :

$$Dir(\alpha_1, \dots, \alpha_K) = \frac{\Gamma(\sum_k \alpha_k)}{\prod_k \Gamma(\alpha_k)} \prod_{k=1}^K p_k^{\alpha_k - 1}$$

- Hyperparameter α_j : 문서 d 에서 토픽 j 가 샘플링 된 횟수에 대한 사전 관찰 count (문서로부터 단어가 실제로 관찰되기 이전의 값)


Latent Dirichlet Allocation의 등장



LDA :
Dirichlet parameter

Variant LDA의 등장

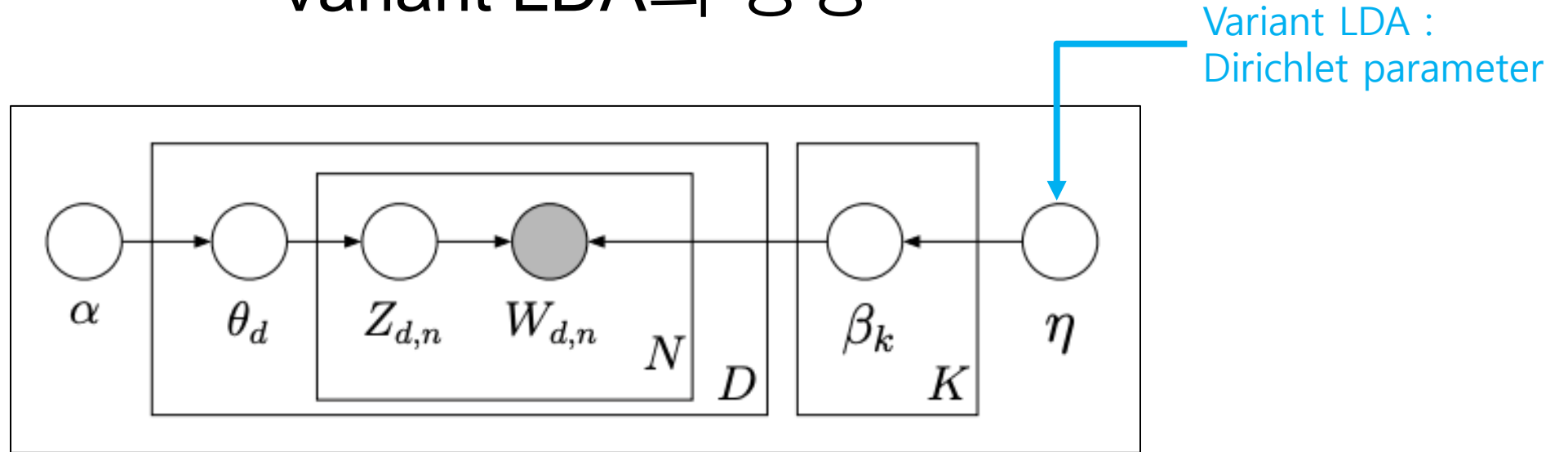
문서 d 의 단어 w_i 대한 분포 :

$$P(w_i) = \sum_{k=1}^K P(w_i | z_i = k) P(z_i = k)$$


Variant LDA는 symmetric Dirichlet distribution(η)을 β 의 prior로 사용
(Griffiths and Steyvers, 2004)

Hyperparameter η : Corpus의 단어가 관찰되기 이전에, 토픽에서 단어가 샘플링 된 횟수에 대한 사전 관찰 count

Variant LDA의 등장



α	Dirichlet parameter		
θ_d	문서 d 에서 토픽 비율(proportion)	$\theta_{d,k}$	문서 d 에서 특정 토픽 k 의 proportion
Z_d	문서 d 에서 토픽 할당(assignment)	$Z_{d,n}$	문서 d 에서 n -th 단어에 대한 토픽 할당
W_d	문서 d 에서 관찰된 단어들	$W_{d,n}$	문서 d 에서 n -th 단어
β_k	토픽 k 의 vocabulary에서의 분포 (단어 전체 셋에서 정의된 토픽 k 의 분포)	η	Dirichlet parameter
The plate surrounding θ_d		각 문서 d 에 대하여, 토픽 분포의 sampling (총 D 개의 문서)	
The plate surrounding β_k		각 topic k 에 대하여, 단어 분포의 sampling (총 K 개의 토픽)	

LDA 모델의 변수

θ'_d s :

Document	Topic 1	Topic 2	Topic 3	...	Topic K
Document 1 (θ_1)	0.2	0.4	0.0	...	0.1
Document 2 (θ_2)	0.8	0.1	0.0	...	0.0
...
Document M (θ_M)	0.5	0.4	0.1	...	0.0

→ 합 : 1

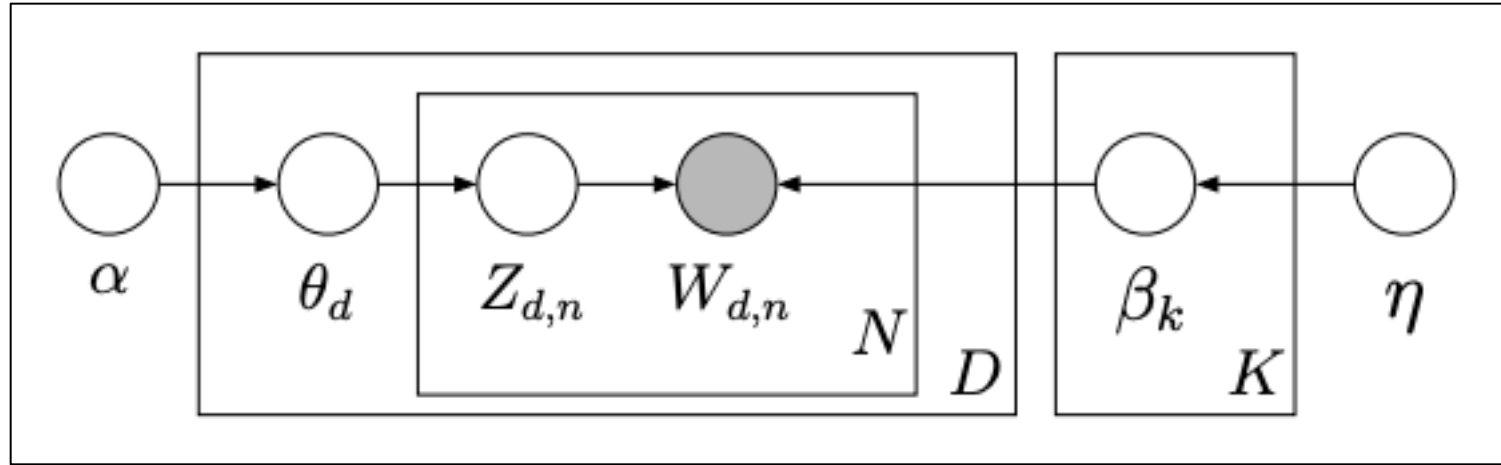
β'_k s :

Terms	Topic 1 (β_1)	Topic 2 (β_2)	Topic 3 (β_3)	...	Topic K (β_K)
Word 1	0.02	0.09	0.00	...	0.00
Word 2	0.08	0.52	0.37	...	0.03
...
Wordt V	0.05	0.12	0.01	...	0.45

↓
합 : 1

- Variant LDA version을 사용하고 있으므로, 이 version을 LDA 라고 지칭 하겠음

LDA의 Generative process



LDA는 **generative model**이다.

1. 문서의 단어의 갯수 N 을 Poisson 분포를 이용하여 선택한다. $N \sim \text{Poisson}(\xi)$
2. 문서의 토픽 분포(proportion) θ_d 를 Dirichlet(α) 분포를 이용하여 선택한다. $\theta_d \sim \text{Dirichlet}(\alpha)$
3. 문서의 단어 각각에 대하여
 - a. 토픽 분포 θ_d 를 이용하여, 단어에 토픽을 할당한다. $Z_{d,n} \sim \text{Multinomial}(\theta)$
 - b. $p(W_{d,n}|Z_{d,n}, \beta)$ 를 이용하여 단어를 선택한다. 이 확률분포는 다항분포이다.

LDA의 Generative process

예제)

1. 새로운 문서 D 의 길이를 5로 선택한다. 즉, $D = (w_1, w_2, w_3, w_4, w_5)$
2. 문서 D 의 토픽 분포를 50%는 음식(food), 50%는 동물(animal)로 선택한다.
3. 각 단어에 대하여,
 1. 첫번째 단어 w_1 에 food topic을 할당한다. Food topic에서 broccoli를 w_1 으로 선택한다.
 2. 두번째 단어 w_2 에 animal topic을 할당한다. Animal topic에서 panda를 w_2 으로 선택한다.
 3. 세번째 단어 w_3 에 animal topic을 할당한다. Animal topic에서 adorable 를 w_3 으로 선택한다.
 4. 네번째 단어 w_4 에 food topic을 할당한다. Food topic에서 cherries 를 w_4 으로 선택한다.
 5. 다섯번째 단어 w_5 에 food topic을 할당한다. Food topic에서 eating 를 w_5 으로 선택한다.



D : broccoli panda adorable cherries eating

LDA 모델의 inference

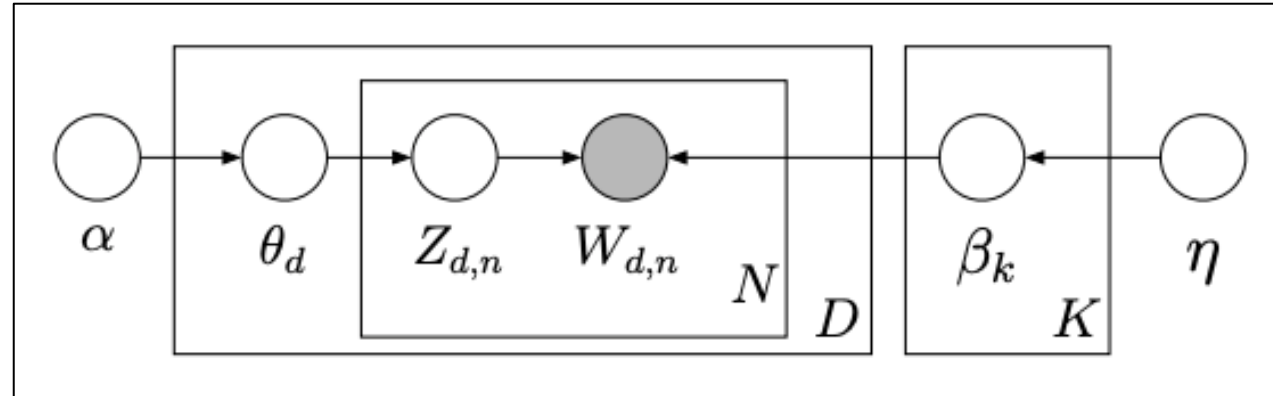
관찰 가능한 문서 내 단어 $w_{d,n}$ 를 이용하여, LDA 모델의 잠재 변수(hidden variable)인 문서의 토픽분포 θ_d 와 토픽의 단어분포 β_k 를 추정하는 과정이 inference이다.

Generative probabilistic modeling에서는, data는 잠재 변수(hidden variable)를 포함하는 generative process에서부터 발생하는것으로 다룬다. 따라서, generative process는 observed random variable과 hidden random variable의 결합 확률밀도(joint probability distribution)를 정의한다.

- Observed variables : 문서내의 단어들
- Hidden variables : 문서의 토픽 분포, 토픽의 단어 분포 (topic structure)

결합 확률밀도함수를 이용하여 observed variable이 주어졌을 때 hidden variable의 조건부 분포를 구한다. 이 분포는 사후 확률분포(posterior distribution)이다.

LDA 모델의 inference



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

관찰 가능 데이터 $w_{1:D}$ 를 통해서 inference해야 할 변수 : $\beta_{1:D}, \theta_{1:D}, z_{1:D}$

Posterior dist.

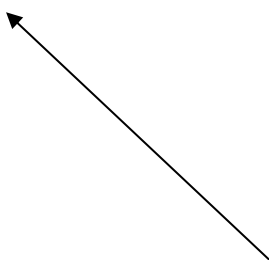
$$p(\beta_{1:K}, \theta_{1:D}, z_{1:D} | w_{1:D}) = \frac{p(\beta_{1:K}, \theta_{1:D}, z_{1:D}, w_{1:D})}{p(w_{1:D})}$$

$\beta_{1:K}$	토픽 1~K의 vocabulary에서의 분포
$\theta_{1:D}$	문서 1~D에서의 토픽 비율
$z_{1:D}$	문서 1~D에서의 토픽 할당
$w_{1:D}$	문서 1~D에서 관찰된 단어들

LDA 모델의 inference

Posterior distribution을 구하는 것은 쉬운것인가?

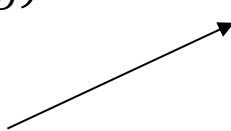
분자의 경우를 먼저 살펴보자.

$$p(\beta_{1:K}, \theta_{1:D}, z_{1:D} | w_{1:D}) = \frac{p(\beta_{1:K}, \theta_{1:D}, z_{1:D}, w_{1:D})}{p(w_{1:D})}$$


모든 random variable의 결합 확률밀도 함수는,
hidden variable이 임의로 셋팅된다면 쉽게 계산 가능

LDA 모델의 inference

분모의 경우를 살펴보자.

$$p(\beta_{1:K}, \theta_{1:D}, z_{1:D} | w_{1:D}) = \frac{p(\beta_{1:K}, \theta_{1:D}, z_{1:D}, w_{1:D})}{p(w_{1:D})}$$



Observed variable의 주변 밀도함수(marginal probability)

- 임의의 topic model에서, observed corpus를 볼 수 있을 확률을 구하는 것
- 모든 hidden topic structure의 가능한 경우(instantiation)를 구하고, 결합 확률밀도 함수를 summation

가능한 hidden topic structure는 지수적으로 많다. 따라서 해당 밀도함수를 구하는 것은 매우 어렵다.

Modern probabilistic models, Bayesian statistics에서는 분모의 분포 때문에 posterior를 계산하는 것이 어렵다. 따라서 posterior를 효과적으로 추정하는 기법에 대한 연구가 많이 이루어지고 있다.

따라서, topic modeling algorithms에서도 **posterior distribution을 추정하기 위한 기법**을 활용한다.

- sampling based method
 - variational method
- 

Difficulty of deriving marginal probability $p(w_{1:D})$

- Topic mixture $\theta \models$ joint distribution (parameter : α, β)

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

- Document \models marginal distribution

$$p(\mathbf{w} | \alpha, \beta) = \int p(\theta | \alpha) \left(\prod_{n=1}^N \sum_{z_n} p(z_n | \theta) p(w_n | z_n, \beta) \right) d\theta$$

- Corpus \models probability

$$p(D | \alpha, \beta) = \prod_{d=1}^M \int p(\theta_d | \alpha) \left(\prod_{n=1}^{N_d} \sum_{z_{dn}} p(z_{dn} | \theta_d) p(w_{dn} | z_{dn}, \beta) \right) d\theta_d$$

Algorithm for extracting topics

Gibbs sampling

$$P(\mathbf{z}_i = j | \mathbf{z}_{-i}, w_i, d_i, \cdot) \propto \left(\frac{C_{wij}^{VK} + \eta}{\sum_{w=1}^V C_{wj}^{VK} + V\eta} \right) \left(\frac{C_{dij}^{DK} + \alpha}{\sum_{t=1}^K C_{dit}^{DK} + T\alpha} \right)$$

Smoothing

문서에서 i 번째에 나오는 단어 w 의 토픽이 j 일 확률에 영향을 미치는 요소

- 요소 1 : 토픽 j 에 할당된 전체 단어 중에서 해당 단어의 점유율이 높을수록 j 일 확률이 크다.
- 요소 2 : w_i 가 속한 문서 내 다른 단어가 토픽 j 에 많이 할당되었을수록 j 일 확률이 크다.

$z_i = j$	문서에서 i 번째에 나오는 단어 w 에 토픽 j 가 할당
\mathbf{z}_{-i}	i 번째 단어를 제외한 다른 단어들에 대한 토픽 할당
w_i	단어 index
d_i	문서 index
\cdot	다른 정보 및 observed information
C_{wj}^{VK}	단어 w 가 토픽 j 에 할당된 횟수 (현재 i 는 제외)
C_{dj}^{DK}	문서 d 의 단어들 중에서 토픽 j 에 할당된 횟수 (현재 i 는 제외)
η	토픽의 단어 분포 생성에 사용되는 Dirichlet parameter
α	문서의 토픽 분포 생성에 사용되는 Dirichlet parameter

Algorithm for extracting topics

예제)

Doc 0 : ($z_{0,0}, z_{0,1}, z_{0,2}, z_{0,3}$)

Doc 1 : ($z_{1,0}, z_{1,1}, z_{1,2}$)

Doc 2 : ($z_{2,0}, z_{2,1}, z_{2,2}, z_{2,3}$)

Doc 3 : ($z_{3,0}, z_{3,1}, z_{3,2}, z_{3,3}, z_{5,4}$)

Doc 4 : ($z_{4,0}, z_{4,1}, z_{4,2}, z_{4,3}, z_{4,4}$)

Doc 5 : ($z_{5,0}, z_{5,1}, z_{5,2}, z_{5,3}, z_{5,4}, z_{5,5}$)

$z_{i,j}$: i 번째 문서에 j 토픽이 할당된 것을 나타내는 확률변수

1. 확률변수에 랜덤하게 토픽을 할당

2. $z_{0,0}$ 을 제외한 값들을 토대로 $z_{0,0}$ 의 값을 업데이트

3. $z_{0,1}$ 을 제외한 값들을 토대로 $z_{0,1}$ 의 값을 업데이트

....

4. $z_{5,5}$ 을 제외한 값들을 토대로 $z_{5,5}$ 의 값을 업데이트

5. 확률변수가 수렴할 때까지 반복

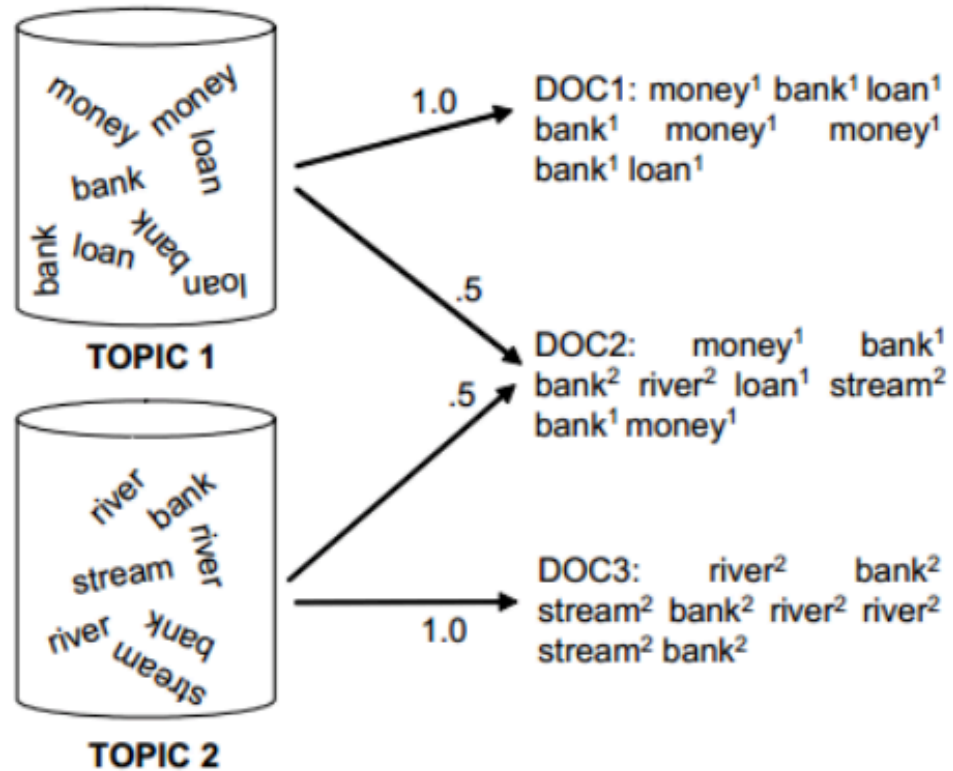
Algorithm for extracting topics

$$C^{VK} = \begin{pmatrix} C_{11} & C_{12} & \dots & C_{1k} & \dots & C_{1K} \\ C_{21} & C_{22} & \dots & C_{2k} & \dots & C_{2K} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ C_{v3} & C_{v3} & \dots & C_{vk} & \dots & C_{vK} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ C_{V1} & C_{V2} & \dots & C_{Vk} & \dots & C_{VK} \end{pmatrix}$$

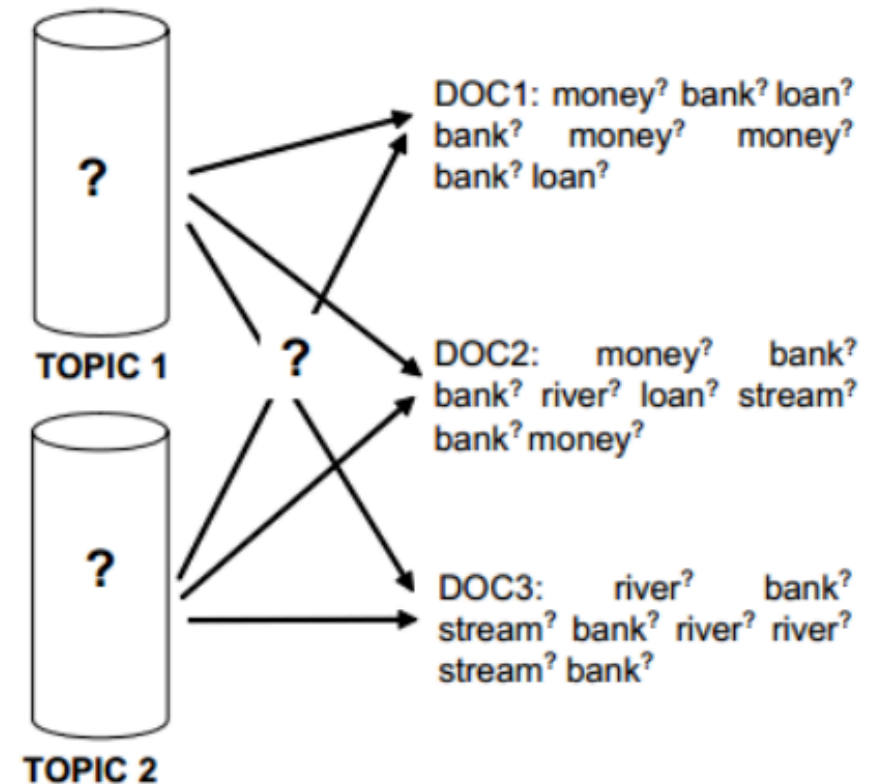
$$C^{DK} = \begin{pmatrix} C_{11} & C_{12} & \dots & C_{1k} & \dots & C_{1K} \\ C_{21} & C_{22} & \dots & C_{2k} & \dots & C_{2K} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ C_{d3} & C_{d3} & \dots & C_{dk} & \dots & C_{dK} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ C_{D1} & C_{D2} & \dots & C_{Dk} & \dots & C_{DK} \end{pmatrix}$$

Generative model vs. Statistical inference

PROBABILISTIC GENERATIVE PROCESS



STATISTICAL INFERENCE



최적의 토픽 수

Perplexity

- Language modeling에서 주로 컨벤션으로 사용한다.
- 일반적으로 perplexity는 $\exp(H(p))$ 로 표현된다. $H(p)$ 는 p 의 엔트로피를 의미한다.
- LDA에서 추정된 토픽 정보를 이용하여 단어의 발생 확률을 계산하였을 때, 확률값이 높을수록 generative process를 제대로 설명한다고 본다.

$$Perplexity(C) = \exp\left(-\frac{\sum_{d=1}^D \log p(\mathbf{w}_d)}{\sum_{d=1}^D N_d}\right)$$

- $p(\mathbf{w}_d)$: 토픽의 단어분포 정보와 문서내 토픽의 비중 정보의 곱을 이용하여 계산
- $p(\mathbf{w}_d)$ 는 클수록 좋으므로, perplexity는 작을수록 좋다.

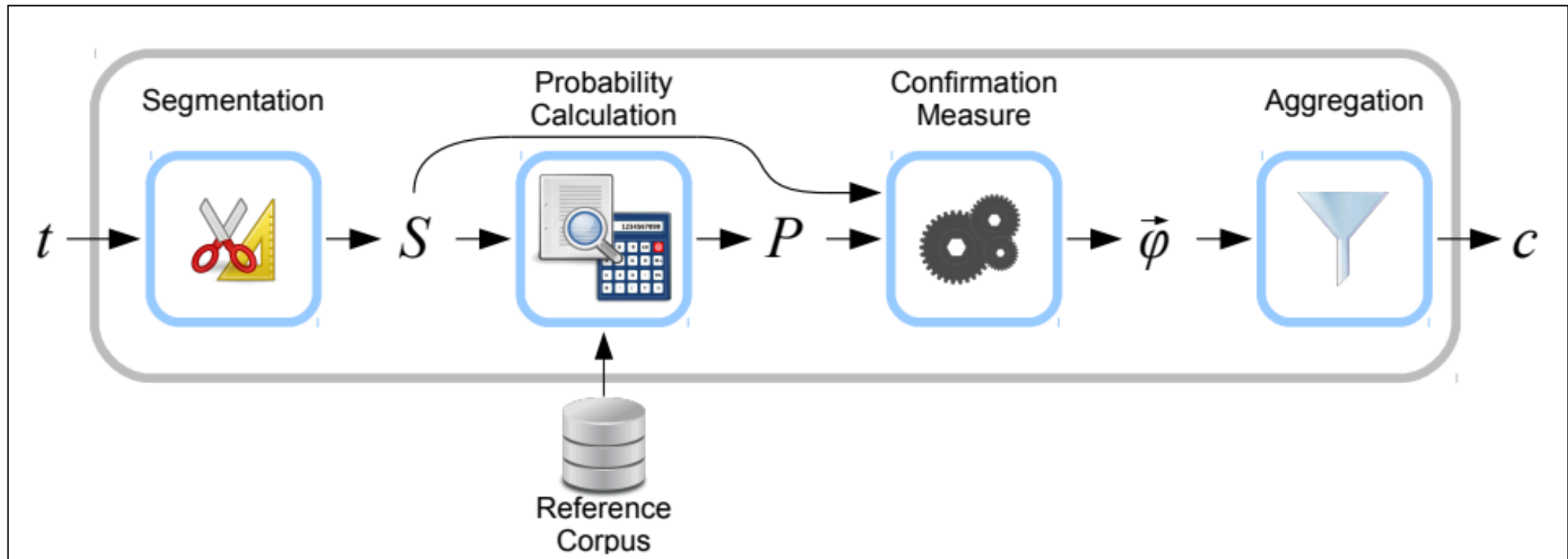
최적의 토픽 수

Topic coherence

- 실제로 사람이 해석하기에(interpretability) 적합한 평가 척도를 만들기 위해 제시된 여러 척도들 중 하나
- Newman은 뉴스와 책 데이터를 수집하여 토픽 모델링을 실시. 그 결과로 나온 토픽들이 유의미한지 수작업으로 점수화. 그리고 이렇게 매겨진 점수와 가장 유사한 결과를 낼 수 있는 척도를 제시.
- 토픽 모델링 결과로 나온 각각의 토픽에서 상위 N 개의 단어를 선택한 후, 상위 단어 간의 유사도를 계산하여, 실제로 해당 토픽이 의미적으로 일치하는 단어들끼리 모여있는지 판단 가능
- 다양한 버전
 - NPMI
 - UMass
 - UCI
 - c_v

최적의 토픽 수

Topic coherence : c_v version



1. Axes of a spatial

- LSA

2. Probabilistic topics

- LDA

3. Bayesian Nonparametric

- HDP

Dirichlet Process

LDA는 토픽의 수 k 가 필요하다. 데이터에 대하여, 이 데이터에 몇 개의 토픽이 존재하는지 미리 아는 것은 어렵다. 이 부분이 LDA의 약점 중 하나이다.

하지만 우리는 데이터에 따라 적절한 토픽 개수를 찾을 수 있으며 이것은 Dirichlet Process를 이용하여 구할 수 있다.

Dirichlet distribution은 주어진 하이퍼파라미터에 따라 다항분포를 생성해주는 분포라고 할 수 있다. 따라서 디리클레 분포를 사전분포로 두면, 다항분포를 따르는 사후확률분포를 쉽게 구할 수 있다. (디리클레 분포는 다항분포의 켄레분포이다.)

Dirichlet Process

X 가 디리클레 분포를 따른다고 가정하자. 즉 $X \sim \text{Dir}(1,2,1)$ 라고 하자. 여기서 $k = 3$ 이다. 따라서 이 분포에서 표본을 추출한다면 3개의 성분으로만 이루어진, 그리고 원소의 합이 1인 샘플들이 생성될 것이다.

$$\mathbb{x}_1 = (0.2, 0.5, 0.3)$$

$$\mathbb{x}_2 = (0.1, 0.6, 0.3)$$

...

디리클레 분포를 서로 다른 k 값이 나올 수 있는 분포로 확장을 한 것이 다음의 **Dirichlet Process**이다.

$$X \sim DP(\alpha, H)$$

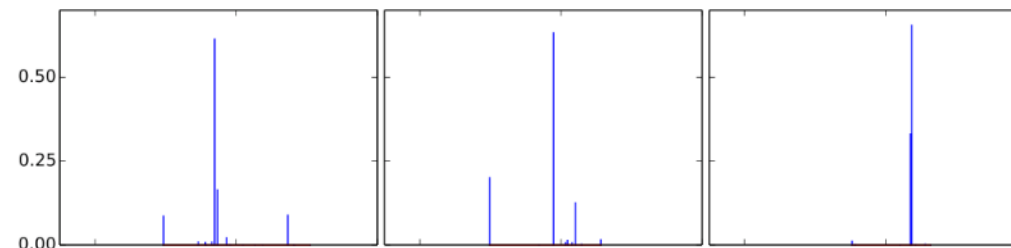
α 는 집중 파라미터(concentration parameter)이고 H 는 모분포이다. α 가 0에 가까울수록 X 의 분포는 모분포의 평균을 중심으로 모이고, α 가 커질수록 모분포의 평균에서 멀어지게 된다.

Dirichlet Process

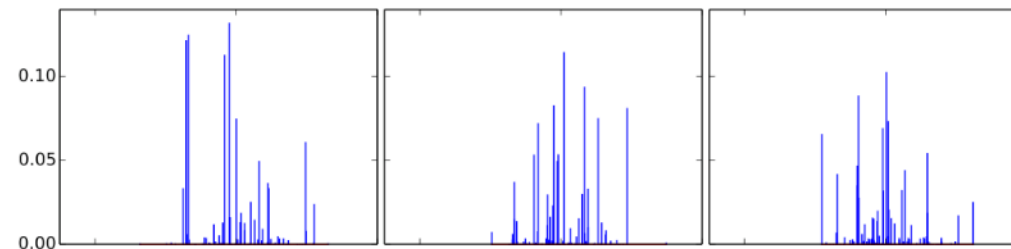
Concentration parameter α 를 변화시켜 얻은,
 X 의 분포에서 추출된 샘플들이다.

표본에 포함된 k (막대 개수)는 같은 α 라도
다르다.

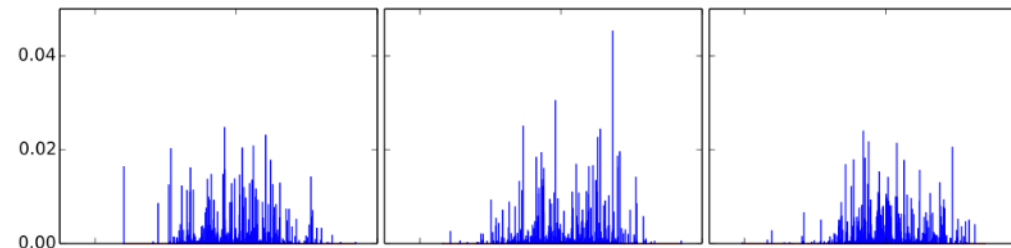
$\alpha = 1$



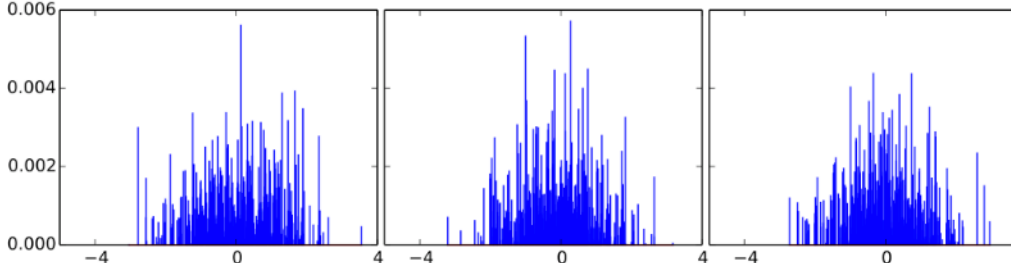
$\alpha = 10$



$\alpha = 100$



$\alpha = 1000$



Chinese Restaurant Process

한 중국집이 있고, 그 중국집에는 무수히 많은 식탁이 있다. 식탁에는 무수히 많은 자리가 있어서 손님이 얼마든지 앉을 수 있다.

단, 손님이 식탁에 앉을 때 아래와 같은 규칙이 있다.

- 손님은 식탁의 인기를 고려해서 어느 식탁에 앉을지 선택한다.
- 식탁에 어떤 음식이 올려질지는 식탁의 첫 손님이 앉을 때 모분포 H 에서 음식 하나를 뽑음으로서 결정한다.
- 식탁의 인기는 앉아 있는 사람의 수에 비례하고, 손님은 비어있는 식탁을 고를 수 있다. 빈 식탁의 인기는 집중 파라미터 α 와 비례한다.

첫번째 손님은 비어있는 식탁에 앉을 것이다. 두번째 손님은 첫번째 식탁에 앉거나, 비어있는 새로운 식탁에 앉는다. 만약 α 가 크다면 비어있는 식탁을 고를 확률이 높아진다. (또는 식탁의 인기가 낮아도)

이렇게 손님이 무한히 계속 들어오다보면 식탁의 개수가 정해지고(countably infinite), 식탁의 인기도 비율도 일정 값에 수렴하게 된다. 이렇게 얻은 인기도의 비는 모분포가 H , 집중 파라미터가 α 인 디리클레 프로세스에서 뽑은 샘플이 된다.

Hierarchical Dirichlet Process

DP의 애로사항 : 만약 중국집이 한 곳이 아니고 여러곳이라고 생각하자. 어떤 음식이 올려진 식탁을 찾아서 각각의 중국집에서 그 음식이 얼마나 인기 있는지 확인하고 싶다고 하자. 문제는 여기서 발생한다. 각 중국집에 식탁이 몇 개 있는지 모르고, 각 식탁에 어떤 음식이 있는지 모른다. 어떤 음식이 서로 다른 식당에서 같이 등장한다는 보장도 없다. 식당 간 음식 비교가 불가능하다.

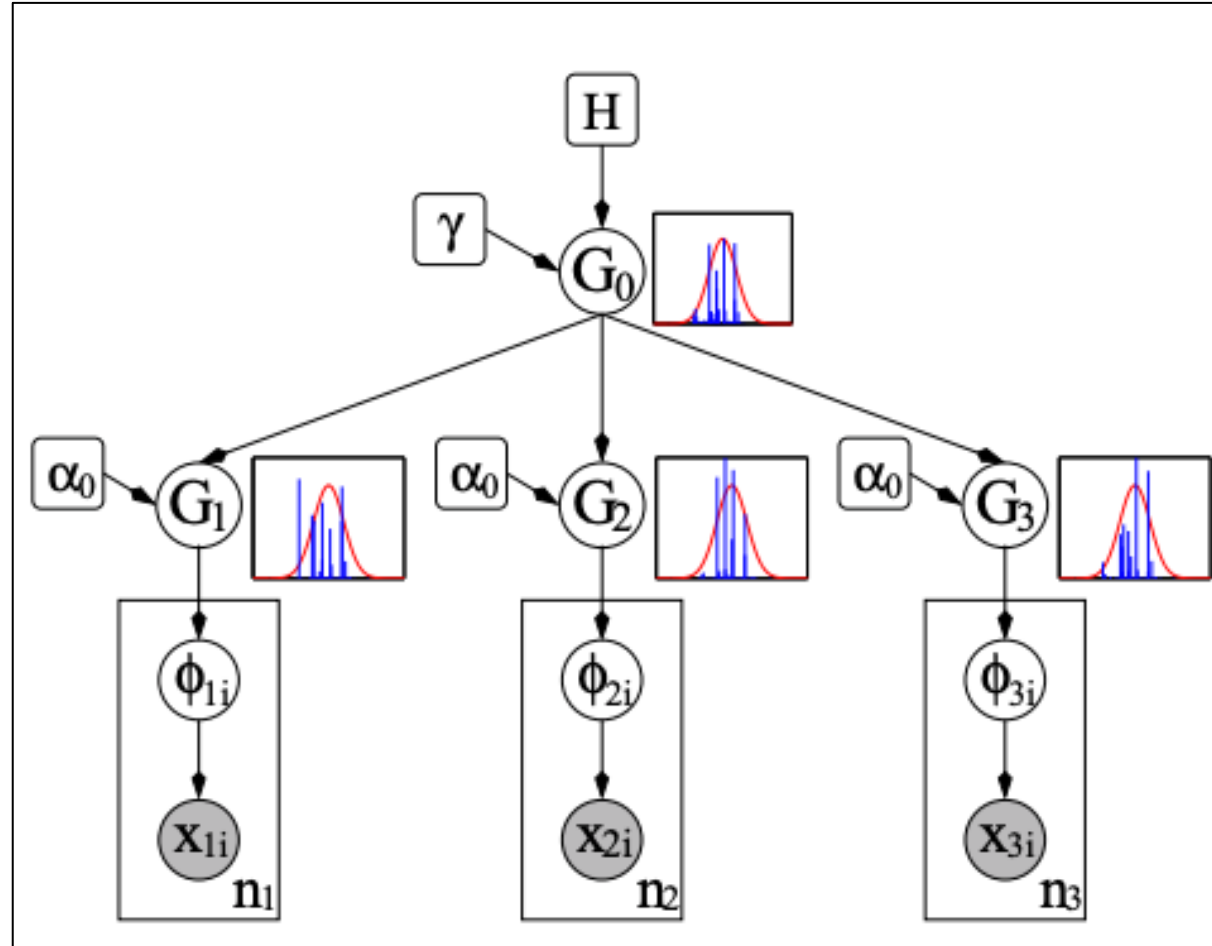
그래서 학자들은 Hierarchical Dirichlet Process를 제안했다.

$$G_0 \sim DP(\gamma, H)$$

$$G_j \sim DP(\alpha, G_0)$$

모분포를 따르는 상위 디리클레 프로세스를 생성하고, 이 디리클레 프로세스를 모분포로 하는 하위 디리클레 프로세스를 여러 개 생성한다. 이런식으로 하위 디리클레 프로세스를 서로 연결 시켜준다.

Hierarchical Dirichlet Process



CRP using HDP

중국집 체인점이 있다. 체인점에는 무수히 많은 식탁이 있고, 그 식탁에는 무수히 많은 자리가 있다. 각 체인점마다 손님이 한명씩 꾸준히 들어와서 식탁에 골라 앉는다. 각각의 체인점에 음식을 배분해주는 중국집 본사가 있다. 각 체인점에 손님이 빈 식탁에 앉게 되면, 직원이 본사로 가서 식탁에 올릴 음식을 골라온다. 본사에는 음식이 담겨있는 아주 큰 접시가 무한히 있고, 각 접시는 충분히 커서 무한히 음식을 풀 수 있다.

CRP using HDP

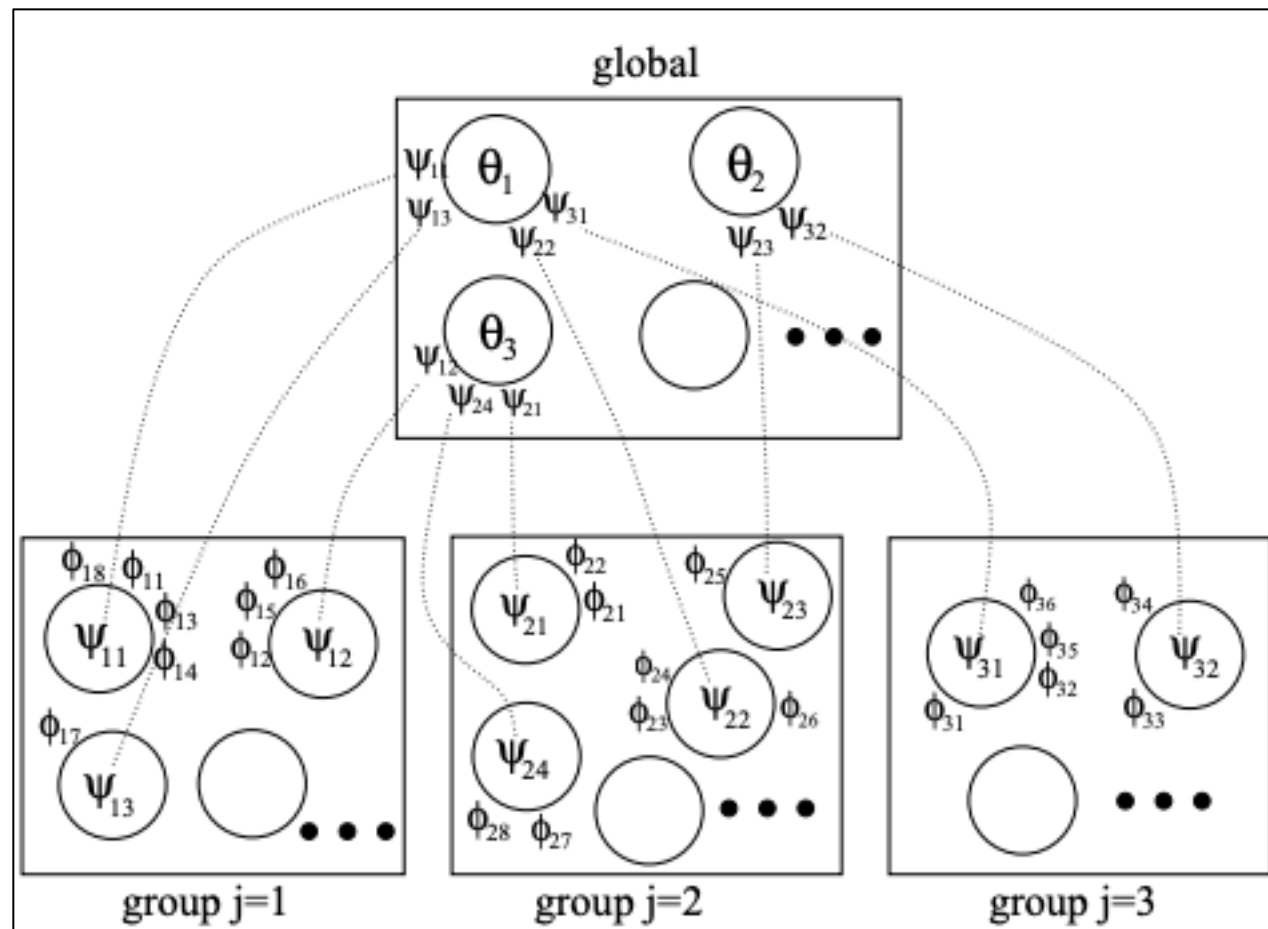
$[G_0 \sim DP(\gamma, H)]$

- 직원은 접시의 인기를 고려해서 어느 접시에서 음식을 풀지 결정
- 접시에 어떤 음식이 올려질지는 접시에서 첫 직원이 음식을 풀 때, 모분포 H 에서 음식을 하나 뽑음으로 결정
- 접시의 인기는 음식을 푸는 직원의 수에 비례하고, 직원이 비어있는 접시를 고를 수도 있다. 접시의 인기는 집중 파라미터 γ 와 상관있다.

$[G_j \sim DP(\alpha, G_0)]$

- 손님은 식탁의 인기를 고려해서 어느 식탁에 앉을지를 고른다.
- 식탁에 어떤 음식이 올려질지는 식탁에 첫 손님이 앉으면 직원을 본사로 보내서 결정하고, 그 음식의 분포 역시 직원들이 접시에서 음식을 푸는 분포 G_0 를 따른다. (식탁에는 1개의 음식만 올라간다)
- 식탁의 인기는 앉아 있는 사람의 수에 비례하고, 집중 파라미터 α 에 따라서 비어있는 식탁을 고를수도 있다.

CRP using HDP



HDP for infinite topic models

- 중국집 체인점 – Documents
- 본사에서 결정해주는 음식 – Topic
- 각각의 손님들 – Words in a document

