문장 수준 임베딩

5-1~5-3

전찬웅

잠재 의미 분석

- 단어 수준 임베딩 (잠재 의미 분석)

  단어-문서, TF-IDF, 단어-문맥, PMI 행렬 → SVD → 단어 벡터 추출

잠재 의미 분석

- 단어 수준 임베딩 (잠재 의미 분석)

  단어-문서, TF-IDF, 단어-문맥, PMI 행렬 → SVD → 단어 벡터 추출

- 문장 수준 임베딩 (잠재 의미 분석)

  단어-문서, TF-IDF → SVD → 문장 벡터 추출

잠재 의미 분석 (예제)

최대엔트로피모델 파라메터 추정

29 Oct 2017 | Maximum Entropy Model

이번 글에서는 최대엔트로피모델(Maximum Entropy model)의 파라메터 추정을 살펴보도록 하겠습니다. 이 글은 기본적으로 이곳을 참고하였습니다. 그럼 시작하겠습니다.

모델 정의

최대엔트로피 모델은 다음과 같이 정의됩니다.

$$P_\Lambda(y|x) = \frac{exp(\sum_i \lambda_i f_i(x,y))}{\sum_y exp(\sum_i \lambda_i f_i(x,y))}$$

위 식에서 f는 x와 y가 주어졌을 때 0 또는 1의 값을 반환하는 함수이며, $f_i(x,y)$는 자질벡터의 i번째 값을 나타냅니다. λ는 $f_i(x,y)$가 얼마나 중요한지 나타내는 가중치입니다. Λ는 그 요소가 $\lambda_1, \lambda_2, \ldots, \lambda_n$인 가중치 벡터입니다.

최대우도추정

최대엔트로피모델 P_Λ에 대한 실제 데이터 분포(empirical distribution) $\tilde{p}(x,y)$의 로그우도 함수는 다음과 같이 정의됩니다.

$$L_{\tilde{p}} = \sum_{x,y} \tilde{p}(x,y) \log P_\Lambda(y|x)$$

자질벡터와 데이터가 주어졌을 때 위 로그우도 함수를 최대화하는 파라메터 Λ를 찾는 것이 목적이 됩니다. $L_{\tilde{p}}$는 1차 도함수가 0인 지점에서 극값을 가지므로, 우리가 구하고자 하는 미지수인 λ_i에 대해 각각 편미분을 하여 정리하면 다음과 같습니다.

잠재 의미 분석(예제)(preprocessing → TF-IDF → SVD → 문장 벡터 추출)

maxparam 이번 글에서는 최대 엔트로피모델(Maximum Entropy model)의 파라미터 추정을 살펴보도록 하겠습니다. 이 글은 기본적으로 이곳을 참고하였습니다. 그럼 시작하겠습니다. ## 모델 정의 최대 엔트로피 모델은 다음과 같이 정의됩니다. $$ { P }{ \Lambda }(y|x)=\frac { { exp(}\sum { i }^{ }{ { \lambda } { i }{ f }{ i }\left(x,y \right) }) } { \sum { y }^{ }{ { exp(}\sum { i }^{ }{ { \lambda }{ i }{ f }{ i }\left(x,y \right) }) } } $$ 위 식에서 f는 x와 y가 주어졌을 때 0 또는 1의 값을 반환하는 함수이며, $f_i(x,y)$는 자질벡터의 i번째 값을 나타냅니다. $λ$는 $f_i(x,y)$가 얼마나 중요한지 나타내는 가중치입니다. $ʌ$는 그 요소가 $\{λ_1, λ_2,...,λ_n\}$인 가중치 벡터입니다. (하략)

```
from preprocess import get_tokenizer

corpus_fname = "data/processed/processed_blog.txt"

tokenizer = get_tokenizer("mecab")
titles, raw_corpus, noun_corpus = [], [], []
with open(corpus_fname, 'r', encoding='utf-8') as f:
    for line in f:
        try:
            title, document = line.strip().split("\u241E")
            titles.append(title)
            raw_corpus.append(document)

            nouns = tokenizer.nouns(document)
            noun_corpus.append(' '.join(nouns))
        except:
            continue
```

이번 글 최대 엔트로피 모델 파라 미터 추정 글 기본 이곳 참고 시작 모델 정의 최대 엔트로피 모델 다음 정의 위 식 때 값 반환 함수 자질 벡터 번 값 중요 가중치 요소 가중치 벡터 (하략)

잠재 의미 분석(예제)(preprocessing → TF-IDF → SVD → 문장 벡터 추출)

이번 글 최대 엔트로피 모델 파라 미터 추정 글 기본 이곳 참고 시작 모델 정의 최대 엔트로피 모델 다음 정의 위 식 때 값 반환 함수 자질 벡터 번 값 중요 가중치 요소 가중치 벡터 (하략)

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(
    min_df=1,
    ngram_range=(1, 1),
    lowercase=True,
    tokenizer=lambda x: x.split())
input_matrix = vectorizer.fit_transform(noun_corpus)
id2vocab = {vectorizer.vocabulary_[token]:token
            for token in vectorizer.vocabulary_.keys()}
# curr_doc : 말뭉치 첫 번째 문서의 TF-IDF 벡터
curr_doc, result = input_matrix[0], []
# curr_doc에서 TF-IDF값이 0이 아닌 요소들을 내림차순 정렬
for idx, el in zip(curr_doc.indices, curr_doc.data):
    result.append((id2vocab[idx], el))
sorted(result, key=lambda x: x[1], reverse=True)
```

[('우도', 0.3093543375424738),
('최대', 0.264419726900156),
('모델', 0.2150954393031573),
('엔트로피', 0.20954601175351922),
('디언', 0.20954601175351922),
(하략)

잠재 의미 분석(예제)(preprocessing → TF-IDF → SVD → 문장 벡터 추출)

문서 제목	상위 단어
최대 엔트로피 파라미터 추정	우도, 최대, 모델
Word Weighting, TF-IDF	단어, 문서, 등장
한국어 서술어의 논항과 자릿수	논항, 서술어, 보충어

잠재 의미 분석(예제)(preprocessing → TF-IDF → SVD → 문장 벡터 추출)

문서 제목	상위 단어
최대 엔트로피 파라미터 추정	우도, 최대, 모델
Word Weighting, TF-IDF	단어, 문서, 등장
한국어 서술어의 논항과 자릿수	논항, 서술어, 보충어

```
from sklearn.decomposition import TruncatedSVD
svd = TruncatedSVD(n_components=100)
vecs = svd.fit_transform(input_matrix)
from models.sent_eval import LSAEvaluator
model = LSAEvaluator("data/sentence-embeddings/lsa-tfidf/lsa-tfidf.vecs")
model.most_similar(doc_id=0)
```

Related Posts

최대우도추정(Maximum Likelihood Estimation) 23 Sep 2017

딥러닝 모델의 손실함수 24 Sep 2017

Conditional Random Fields 10 Nov 2017

Variational AutoEncoder 27 Jan 2018

unsupervised generative models 18 Dec 2017

```
[('loss', 0.7452402227234523),
 ('MLE', 0.7254419775507931),
 ('CRF', 0.6840929766260055),
 ('unsugen', 0.6101171993730251),
 ('logistic', 0.5870899037852089),
```

Doc2Vec

- 구글 연구 팀이 개발한 문서 임베딩 기법.

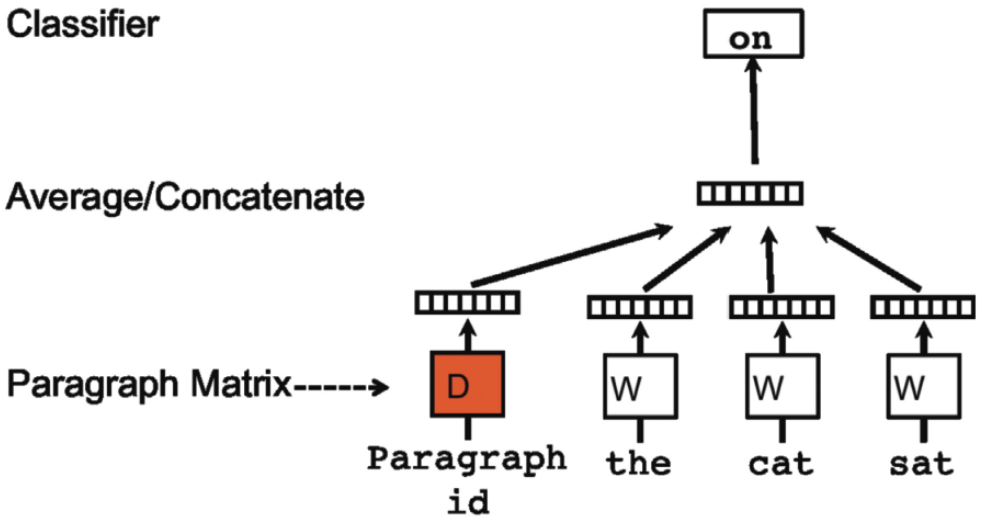


그림 5-6 Doc2Vec PV-DM(Le&Mikolov, 2014)

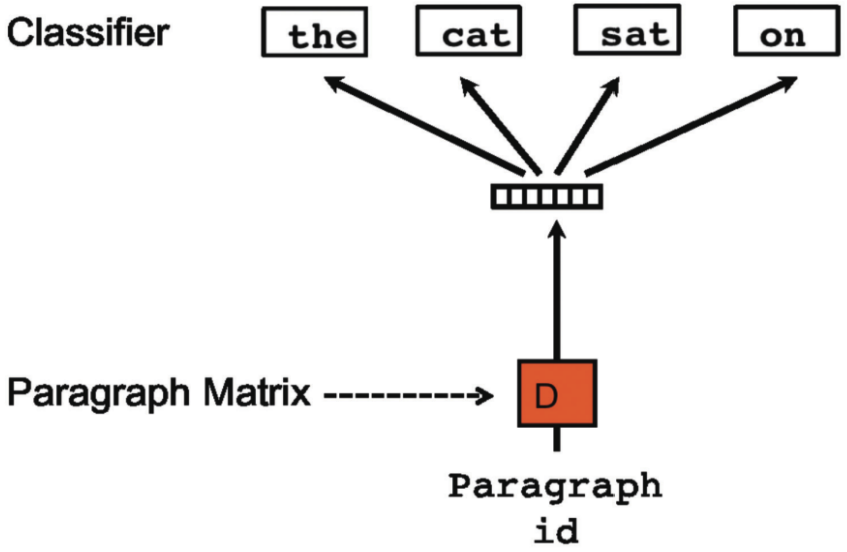
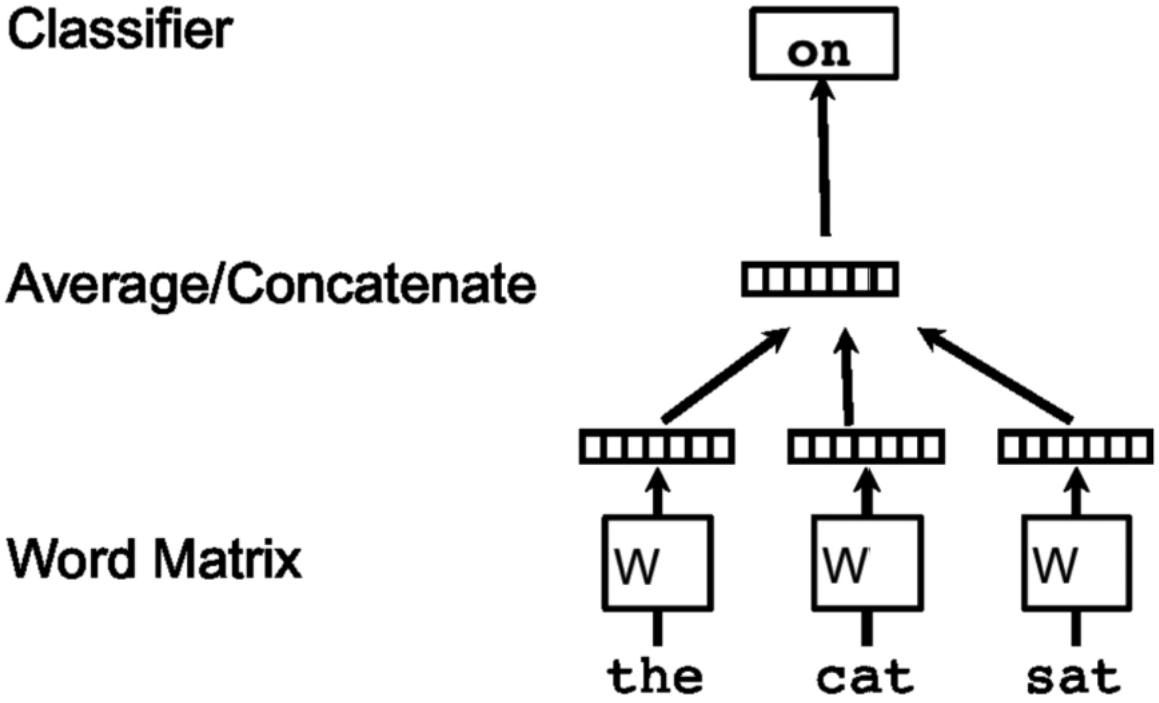


그림 5-7 Doc2Vec PV-DBOW(Le&Mikolov, 2014)

Doc2Vec



$$\mathcal{L} = \frac{1}{T}\sum_{t=k}^{T-1} \log p\left(w_t | w_{t-k}, ..., w_{t-1}\right)$$

$$P\left(w_t | w_{t-k}, ..., w_{t-1}\right) = \frac{\exp\left(\mathbf{y}_{w_t}\right)}{\sum_i \exp\left(\mathbf{y}_i\right)}$$

$$\mathbf{y} = \mathbf{b} + \mathbf{U} \cdot \mathrm{h}\left(w_{t-k}, ..., w_{t-1}; \mathbf{W}\right)$$

Doc2Vec

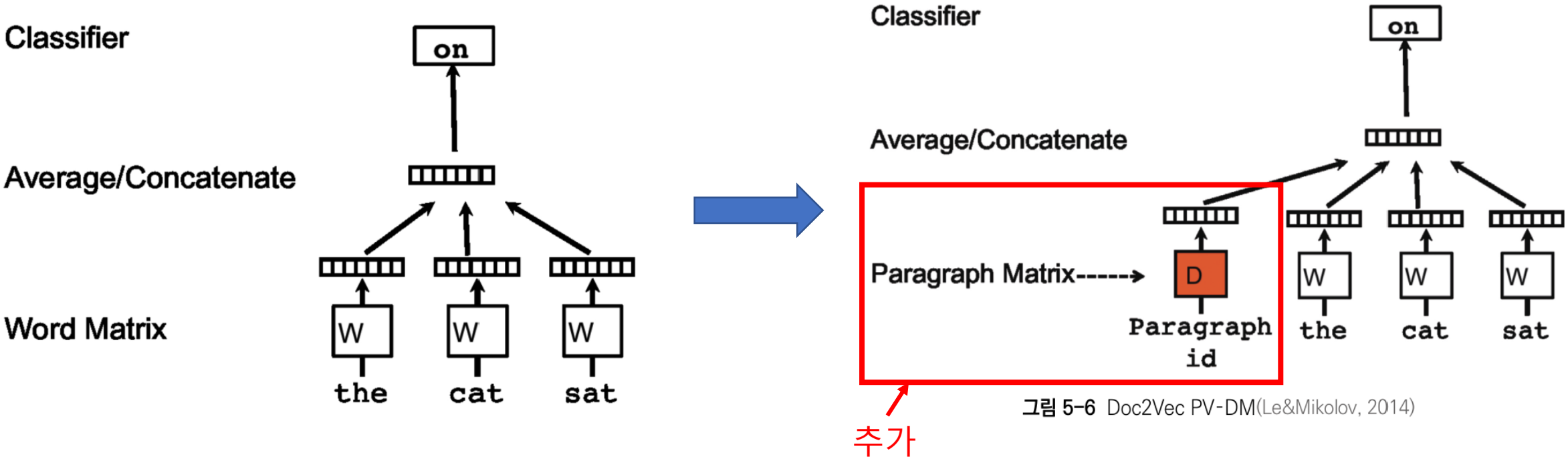


그림 5-6 Doc2Vec PV-DM(Le&Mikolov, 2014)

Doc2Vec

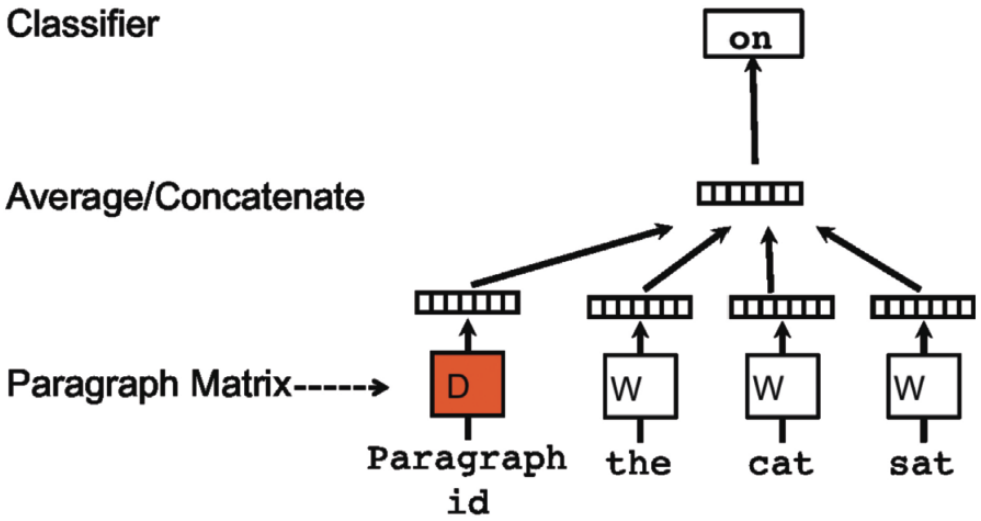


그림 5-6 Doc2Vec PV-DM(Le&Mikolov, 2014)

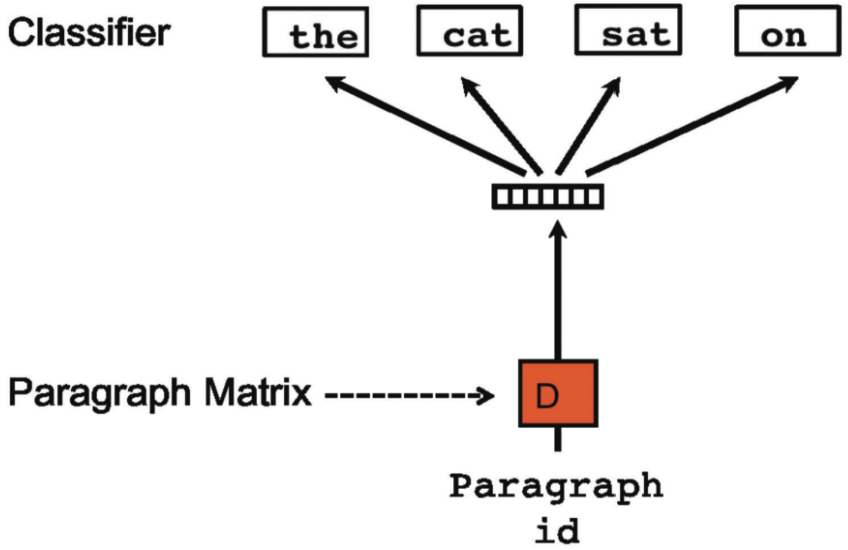
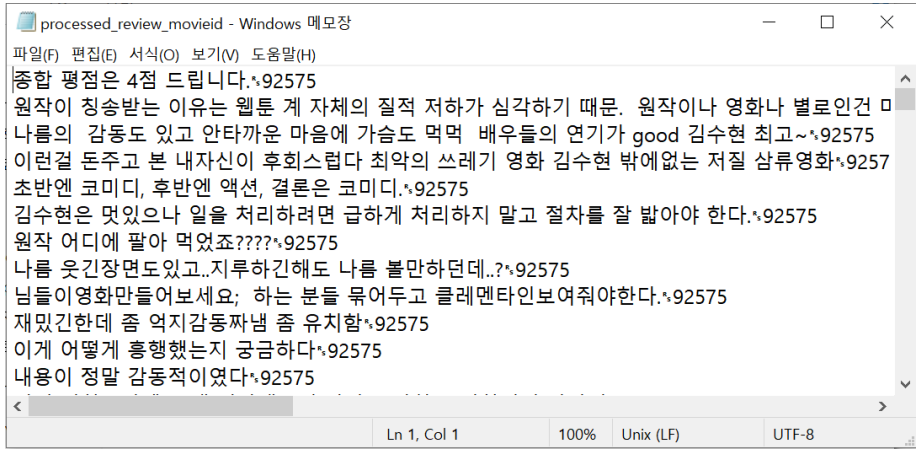


그림 5-7 Doc2Vec PV-DBOW(Le&Mikolov, 2014)

Doc2Vec (예제)



preprocessing

```python
from preprocess import get_tokenizer
from gensim.models.doc2vec import TaggedDocument

class Doc2VecInput:

    def __init__(self, fname, tokenizer_name="mecab"):
        self.fname = fname
        self.tokenizer = get_tokenizer(tokenizer_name)

    def __iter__(self):
        with open(self.fname, encoding='utf-8') as f:
            for line in f:
                try:
                    sentence, movie_id = line.strip().split("\u241E")
                    tokens = self.tokenizer.morphs(sentence)
                    tagged_doc = TaggedDocument(words=tokens,
                                                tags=['MOVIE_%s' % movie_id])
                    yield tagged_doc
                except:
                    continue
```

doc2vec

```python
from gensim.models import Doc2Vec

corpus_fname = "/notebooks/embedding/data/processed/processed_review_movieid.txt"
output_fname = "/notebooks/embedding/data/sentence-embeddings/doc2vec/doc2vec.model"
corpus = Doc2VecInput(corpus_fname)
model = Doc2Vec(corpus, dm=1, vector_size=100)
model.save(output_fname)
```

test

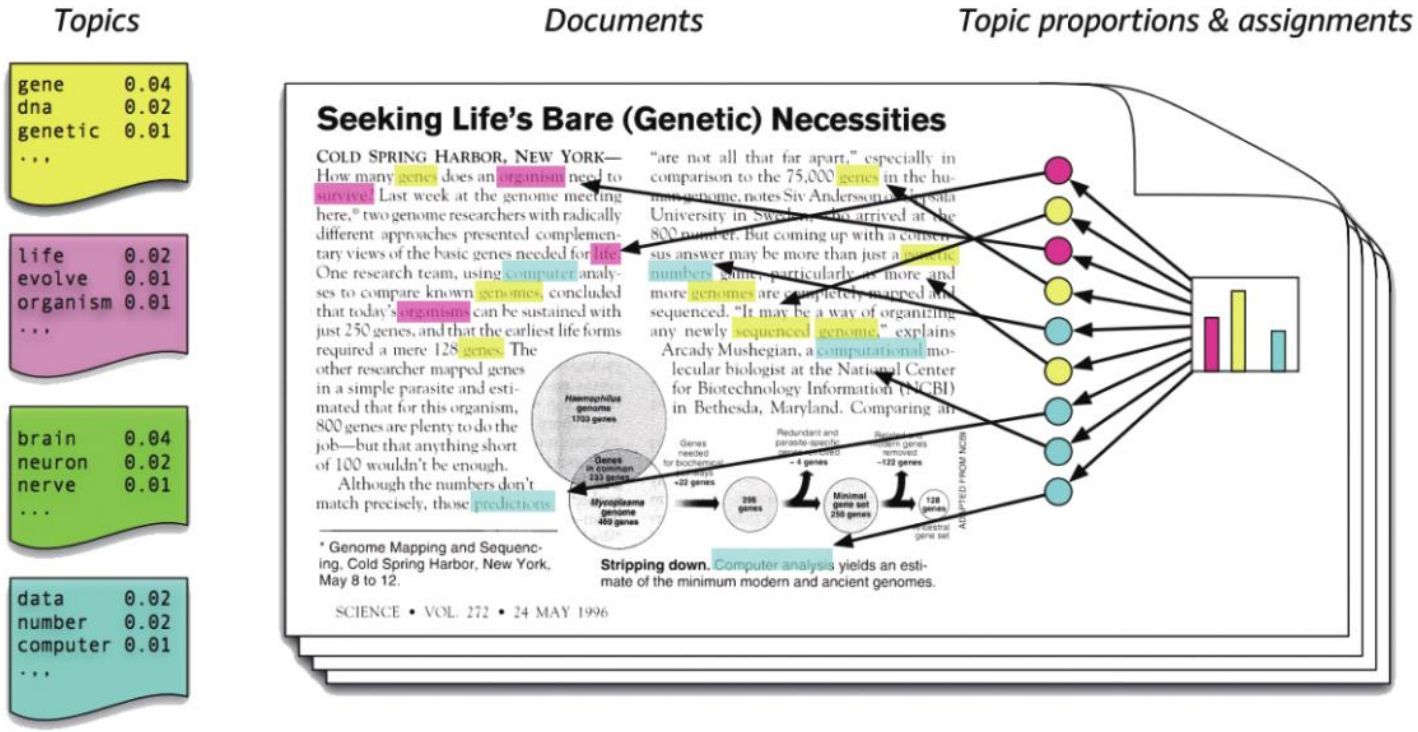
```python
# 36843 : 러브 액츄얼리
model.most_similar(36843, topn=3)
```

result

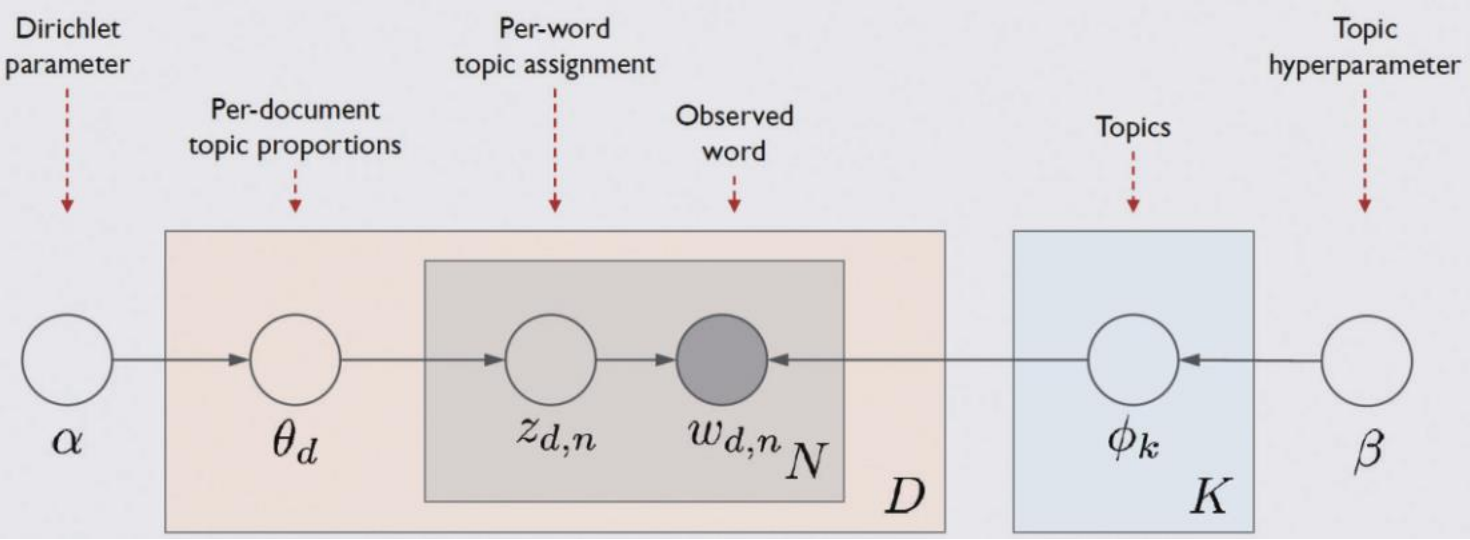
우리, 사랑일까요? 0.7231771349906921
노팅 힐 0.7192620038986206
내 남자의 로맨스 0.7116434574127197

잠재 디리클레 할당

- 주어진 문서에 대해 각 문서에 어떤 주제들이 존재하는지에 대한 확률 모형



잠재 디리클레 할당



D : 말뭉치 전체 문서 개수

K : 전체 토픽 수

N : d번째 문서의 단어 수

(동그라미는 변수 / 네모는 반복횟수를 의미)

디리클레 분포

- k차원의 실수 벡터 중 벡터의 요소가 양수이며 모든 요소를 더한 값이 1인 경우에 확률값이 정의되는 연속확률분포
- 2이상의 자연수 k와 양의 상수 α1,…,αk에 대하여 디리클레분포의 확률밀도함수는 다음과 같다

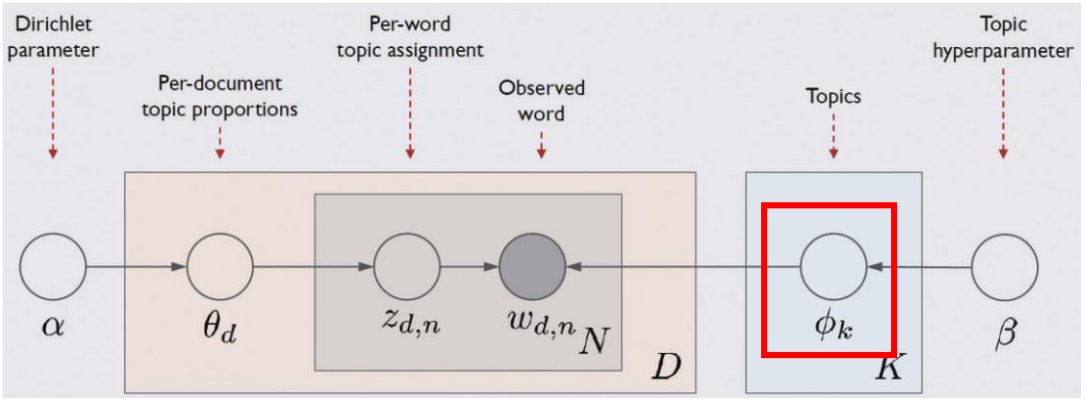
$x_1, \ldots, x_k$가 모두 양의 실수이며 $\sum_{i=1}^{k} x_i = 1$을 만족할 때,

$$f(x_1, \ldots x_k; \alpha_1, \ldots, \alpha_k) = \frac{1}{B(\alpha)} \prod_{i=1}^{k} {x_i}^{\alpha_i - 1}$$

단, $B(\alpha) = \frac{\prod_{i=1}^{k} \Gamma(\alpha_i)}{\Gamma(\sum_{i=1}^{k} \alpha_i)}$

그 외의 경우는 0이다.

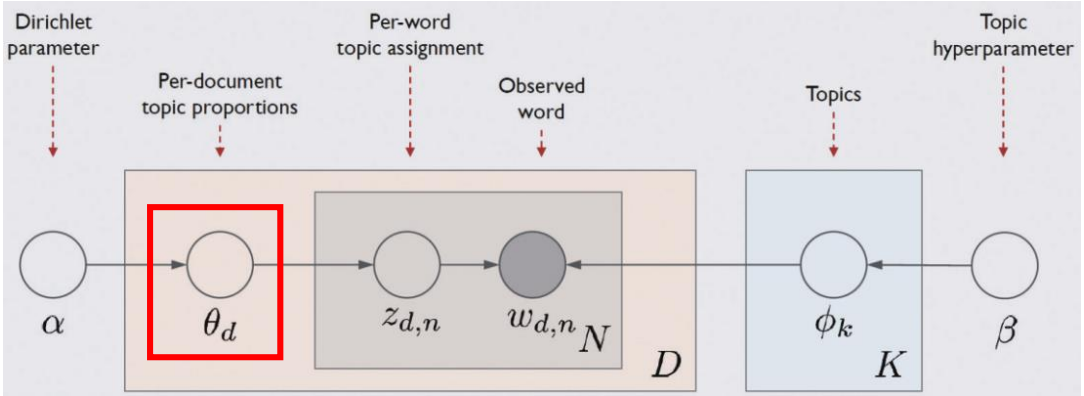
잠재 디리클레 할당



단어	토픽1	토픽2	토픽3
야구	0.000	0.000	0.200
농구	0.000	0.000	0.267
권투	0.000	0.000	0.133
돈	0.231	0.313	0.400
수익	0.000	0.312	0.000
이자율	0.000	0.312	0.000
민주당	0.269	0.000	0.000
자유한국당	0.115	0.000	0.000
전당대회	0.192	0.000	0.000
대통령	0.192	0.063	0.000

ϕ_3 → 3번째 토픽에 해당하는 벡터

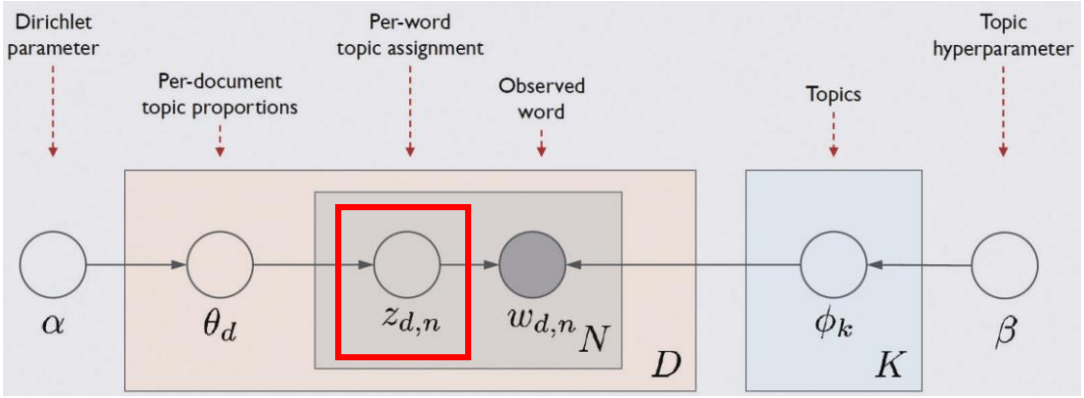
잠재 디리클레 할당



문서	토픽1	토픽2	토픽3
문서1	0.400	0.000	0.600
문서2	0.000	0.600	0.400
문서3	0.375	0.625	0.000
문서4	0.000	0.375	0.625
문서5	0.500	0.000	0.500
문서6	0.500	0.500	0.000

θ_1 → 1번째 문서가 가진 토픽 비중

잠재 디리클레 할당

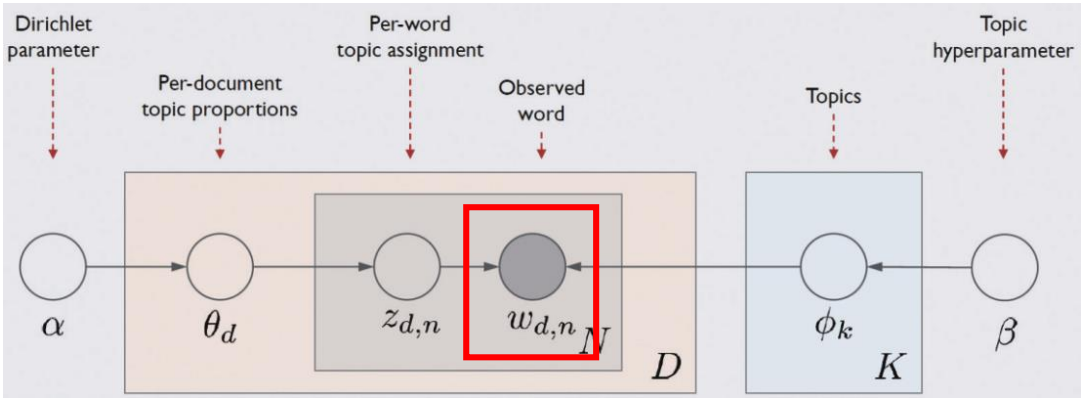


$Z_{d,n}$ 은 d번째 문서 n번째 단어가 어떤 토픽인지를 나타내는 변수

문서	토픽1	토픽2	토픽3
문서1	0.400	0.000	0.600
문서2	0.000	0.600	0.400
문서3	0.375	0.625	0.000
문서4	0.000	0.375	0.625
문서5	0.500	0.000	0.500
문서6	0.500	0.500	0.000

$Z_{3,n}$ 은 토픽2일 가능성이 높다.

잠재 디리클레 할당



$W_{d,n}$은 d번째 문서 내에 n번째로 등장하는 단어

(ex) $Z_{3,1}$이 토픽2라고 할 때 $W_{3,1}$은 돈이 될 가능성이 가장 높다.

단어	토픽1	토픽2	토픽3
야구	0.000	0.000	0.200
농구	0.000	0.000	0.267
권투	0.000	0.000	0.133
돈	0.231	0.313	0.400
수익	0.000	0.312	0.000
이자율	0.000	0.312	0.000
민주당	0.269	0.000	0.000
자유한국당	0.115	0.000	0.000
전당대회	0.192	0.000	0.000
대통령	0.192	0.063	0.000

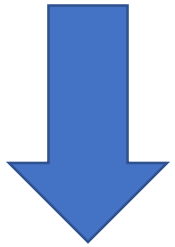
문서	토픽1	토픽2	토픽3
문서1	0.400	0.000	0.600
문서2	0.000	0.600	0.400
문서3	0.375	0.625	0.000
문서4	0.000	0.375	0.625
문서5	0.500	0.000	0.500
문서6	0.500	0.500	0.000

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

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

잠재 디리클레 할당

$$p(z,\ \phi,\ \theta \mid w) = p(z,\ \phi,\ \theta,\ w) / p(w)$$



깁스 샘플링을 활용

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

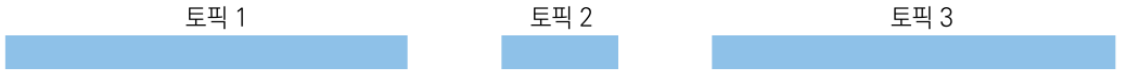
잠재 디리클레 할당

z_{1j}	3	2	1	3	1
$w_{1,n}$	천주교	무역	가격	불교	시장

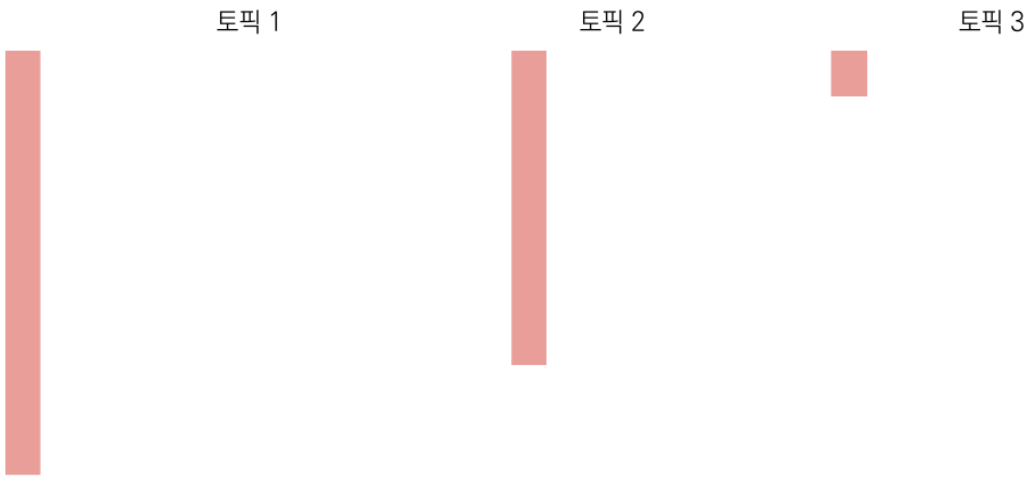
단어	토픽1	토픽2	토픽3
천주교	1	0	35
시장	50	0	1
가격	42	1	0
불교	0	0	20
무역	10	8	1
…	…	…	…

잠재 디리클레 할당

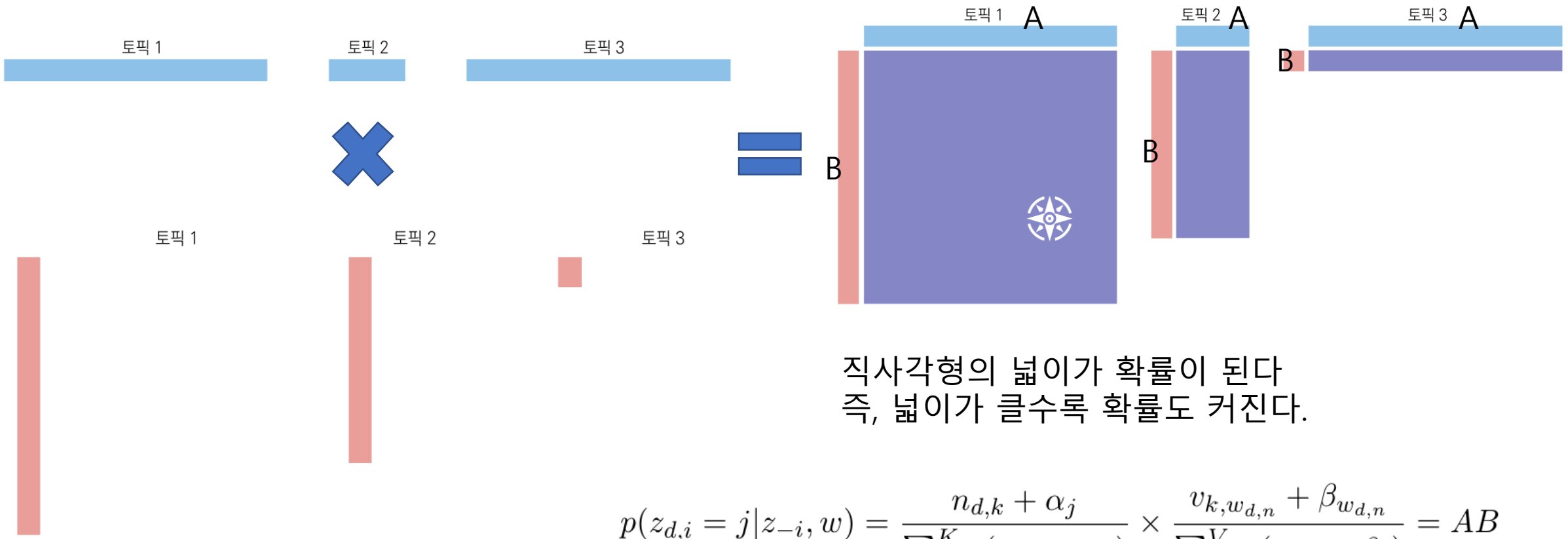
$z_{1,i}$	3	?	1	3	1
$w_{1,n}$	천주교	무역	가격	불교	시장



단어	토픽1	토픽2	토픽3
천주교	1	0	35
시장	50	0	1
가격	42	1	0
불교	0	0	20
무역	10	8-1	1
...	...	...	...



잠재 디리클레 할당



직사각형의 넓이가 확률이 된다
즉, 넓이가 클수록 확률도 커진다.

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

잠재 디리클레 할당(예제)

- https://wikidocs.net/30708 참조