남혜린
우윤희
이가경
조수연

FastText (p. 129~136)

Out of vocabulary, infrequent words (Word2Vec)

영화 (1412516)	관람객 (585858)	재미 (344634)	연기 (255673)	관상 (988)	클로버필드 (136)
애니 (6075)	긋긋 (14681)	제미 (630)	케미 (2257)	광해 (4143)	투모로우 (598)
애니메이션 (7456)	그치만 (1616)	재이 (197)	가창 (104)	베를린 (2441)	다이하드 (277)
작품 (39544)	이지만 (8276)	잼이 (730)	영상미 (11800)	도둑들 (2954)	쿵푸팬더 (94)
명화 (708)	유쾌하고 (2810)	잼 (13098)	목소리 (3489)	역린 (1256)	매트릭스 (928)
드라마 (16306)	but (809)	짜임새 (3739)	캐미 (562)	놈놈놈 (529)	실미도 (337)
에니메이션 (577)	그러나 (9951)	기다린보람이 (98)	아역 (4463)	부당거래 (676)	헝거게임 (121)
엉화 (126)	듯하면서도 (72)	잼미 (120)	카리스마 (3034)	과속스캔들 (850)	레지던트이블 (199)
수작 (5048)	아주 (24571)	ㅈㅐ미 (27)	노래 (24689)	감시자들 (654)	메트릭스 (121)
양화 (164)	다만 (9957)	특색 (164)	열연 (3326)	전우치 (1863)	분노의질주 (194)

클러버필드 (7)	와일더 (5)
characters (5)	짱예 (11)
미라클잼 (5)	생스터 (23)
유웰에 (5)	를라 (13)
디오디오디오디오디오 (8)	존섹 (20)
잡잡잡잡 (5)	윌터너 (39)
내꼬야 (5)	이뻐이뻐 (16)
qf (5)	이뿌구 (13)
굳긋굳긋굳 (5)	77ㅏ (10)
애앰 (6)	긔요미 (19)
romantic (5)	세젤예 (5)

FastText (p. 129~136)

등장배경

Word2Vec의 한계점

1) 모르는 단어 (out of vocabulary) 에 대해서는 word representation vector 를 얻을 수 없다.
2) infrequent words 에 대하여 학습이 불안하다.

FastText 는 이를 보완하기 위하여 Word2Vec 을 제안한 Mikolov 가
2 년 뒤 추가로 제안한 word representation 입니다.

Word2Vec 은 한 단어의 벡터값을 직접 학습하지만,
FastText 는 단어를 구성하는 subwords (substrings) 벡터의 합으로 단어 벡터를 표현합니다.

FASTTEXT (p. 129~136)

모델 기본 구조

```
where -> <wh, whe, her, ere, re>
```

그리고 실제 단어를 나타낼 때에는 3 ~ 6 grams 를 모두 이용합니다.

```
where -> <wh, whe, her, ere, re>
         <whe, wheh, here, ere>
         ...
```

마지막으로, 길이와 상관없이 단어에 <, > 를 더한 subword 는 special unit 으로 추가합니다.

```
where -> <wh, whe, her, ere, re>
         <whe, wheh, here, ere>
         ...
         <where>
```

```
v(where) = v(<wh> + v(whe) + ... v(<where) + v(where>) + v(<where>)
```

FASTTEXT (p. 129~136)

모델 학습

타깃단어(t) , 문맥단어 (c)가 포지티브 샘플일 확률
= t 주변에 c 가 존재

$$p(+|t,c) = \frac{1}{1+\exp(-u_t v_c)} = \frac{1}{1+\exp(-\sum_{g\in\mathcal{G}_w} \mathbf{z}_g^\top \mathbf{v}_c)}$$

타깃단어(t) , 문맥단어 (c)가 네거티브 샘플일 확률
= c를 t와 무관하게 말뭉치 전체에서 랜덤 샘플

$$p(-|t,c) = 1 - p(+|t,c) = \frac{\exp(-\sum_{g\in\mathcal{G}_w} \mathbf{z}_g^\top \mathbf{v}_c)}{1+\exp(-\sum_{g\in\mathcal{G}_w} \mathbf{z}_g^\top \mathbf{v}_c)}$$

* $\mathcal{G}_w \subset \{1,\ldots,G\}$: 타깃 단어 t에 속한 문자단위 n-gram 집합

FASTTEXT (p. 129~136)

모델 학습

로그우도 함수 log-likelihood function

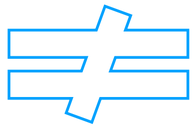
$$L(\theta) = \log P(+|t_p, c_p) + \sum_{i=1}^{k} \log P(-|t_{ni}, c_{ni})$$

FASTTEXT (p. 129~136)

한글 자소와 FastText

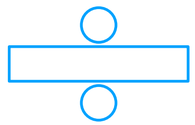
한국어의 오탈자는 초/중/종성에서 한군데 정도가 틀리기 때문에
자음/모음을 풀어서 FastText 를 학습하는게 좋습니다

v(어디야) -> [v(어디), v(디야)]



V(어딛야) -> [v(어딛), v(딛야)]

어디야 -> ‘ㅇㅓ-ㄷㅣ-ㅇㅑ-’



어딛야 -> ‘ㅇㅓ-ㄷㅣㄷㅇㅑ-’

FASTTEXT (p. 129~136)

장점

모르는 단어(Out Of Vocabulary, OOV)에 대한 대응

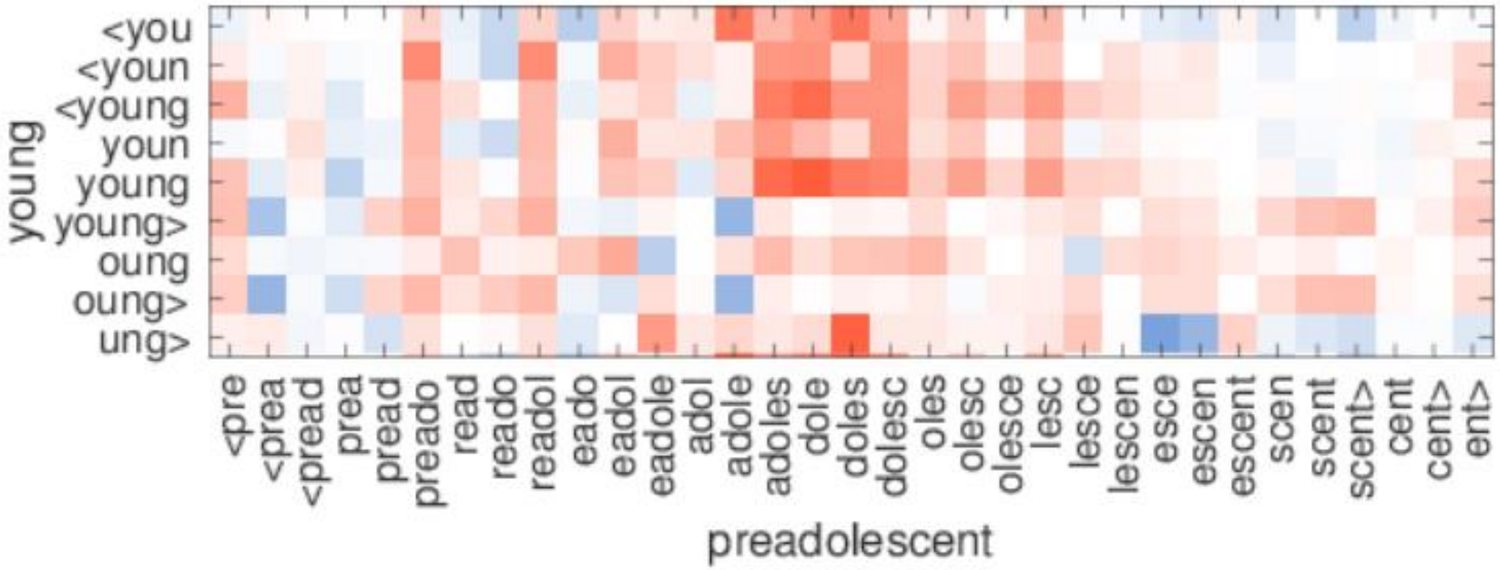
```
Birthplace
Birth + place  -> Birthplace
```

단어 집합 내 빈도 수가 적었던 단어(Rare Word)에 대한 대응

이처럼 단어를 subwords 로 표현하면 typo 에 대하여 비슷한 단어 벡터를 얻을 수 있습니다. Character 3 gram 기준으로 where 와 wherre 는 두 개의 subwords 만 다르고 대부분의 subwords 가 공통으로 존재하기 때문입니다.

```
where -> <wh, whe, her, *ere*, re>
wherre -> <wh, whe, her, *err*, *rre*, re>
```

FASTTEXT (p. 129~136)



FASTTEXT (p. 129~136)

튜토리얼

잠재의미분석(p. 137~142)

잠재의미분석

단어=문서행렬이나 TF-IDF행렬, 단어 문맥 행렬 같은 커다란 행렬에
차원축소의 일종인 특이값 분해를 수행해 차원수를 줄여 계산

행간의 잠재의미를 이끌어 내기 위한 방법.

잠재의미분석(p. 137~142)

PPMI행렬

$$\mathrm{PMI}(x,y) = \log_2 \frac{P(x,y)}{P(x)P(y)}$$

$$\mathrm{PPMI}(x,y) = \max(0, \mathrm{PMI}(x,y))$$

잠재의미분석(p. 137~142)

행렬분해로 이해하는 잠재의미분석

$$\mathbf{X} = \mathbf{U}\mathbf{S}\mathbf{V}^{\mathrm{T}}$$

그림 2-9 SVD에 의한 행렬의 변환(행렬의 '흰 부분'은 원소가 0임을 뜻함)

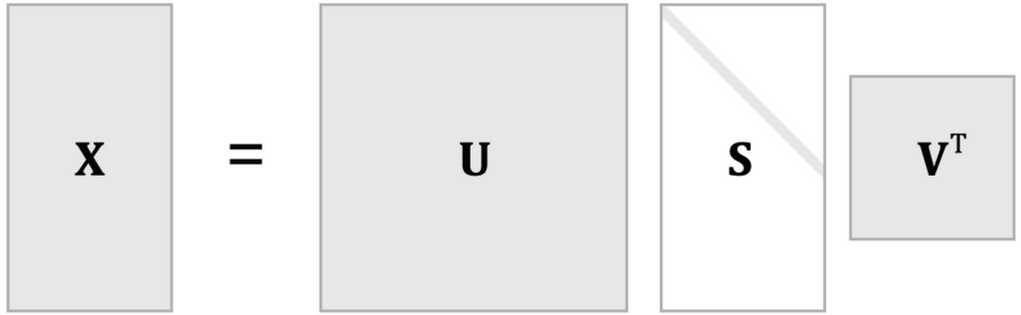


그림 2-10 SVD에 의한 차원 감소

잠재의미분석(p. 137~142)

SVD 예시

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

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

truncated SVD 예시

$$A' = U_1\Sigma_1 V_1^T$$

$$\begin{bmatrix} 1.79 & 4.08 \\ 1.27 & 2.89 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0.82 \\ 0.58 \\ 0 \\ 0 \end{bmatrix} [5.47] [0.40 \quad 0.91]$$

잠재의미분석(p. 137~142)

- pizza
- pizza hamburger cookie
- hamburger
- ramen
- sushi
- ramen sushi

If we use bag of words, pizza and hamburger doesn't have similarity

	pizza	hamburger	cookie	ramen	sushi
pizza	1	0	0	0	0
pizza hamburger cookie	1	1	1	0	0
hamburger	0	1	0	0	0
ramen	0	0	0	1	0
sushi	0	0	0	0	1
ramen sushi	0	0	0	1	1

Similarity(pizza, hamburger) = 0

Similarity(pizza, ramen) = 0

출처: https://www.youtube.com/watch?v=GVPTGq53H5I

잠재의미분석(p. 137~142)

TF-IDF or Bag of Words
similarity is based on word

LSA similarity is based on topic

잠재의미분석(p. 137~142)

Here is word-document matrix

	d1	d2	d3	d4	d5	d6
	pizza	pizza hamburger cookie	hamburger	ramen	sushi	ramen sushi
pizza	1	1	0	0	0	0
ham burger	0	1	1	0	0	0
cookie	0	1	0	0	0	0
ramen	0	0	0	1	0	1
sushi	0	0	0	0	1	1

= A

After SVD matrix decomposition

A ≒ U x Σ x V^T

U

	t1	t2	t3	t4	t5
w1	0.6	0	0	0.7	-0.3
w2	0.6	0	0	-0.7	-0.3
w3	0.5	0	0	0	0.9
w4	0	0.7	-0.7	0	0
w5	0	0.7	0.7	0	0

Word matrix for topic

Σ

	t1	t2	t3	t4	t5	t6
t1	1.9	0	0	0	0	0
t2	0	1.7	0	0	0	0
t3	0	0	1	0	0	0
t4	0	0	0	1	0	0
t5	0	0	0	0	0.5	0

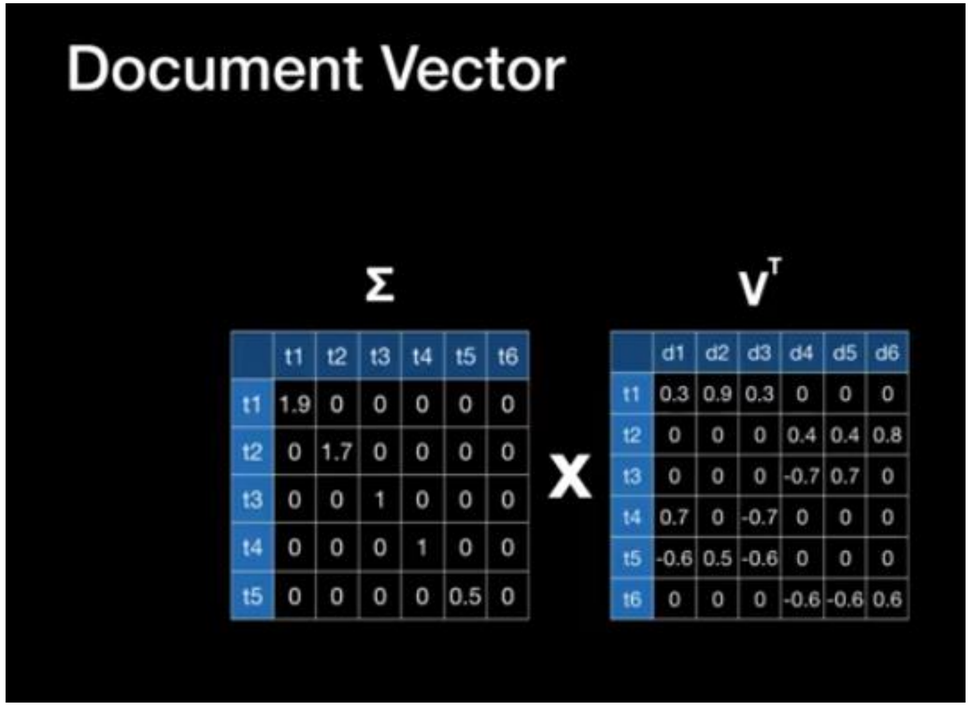
Topic Strength

V^T

	d1	d2	d3	d4	d5	d6
t1	0.3	0.9	0.3	0	0	0
t2	0	0	0	0.4	0.4	0.8
t3	0	0	0	-0.7	0.7	0
t4	0.7	0	-0.7	0	0	0
t5	-0.6	0.5	-0.6	0	0	0
t6	0	0	0	-0.6	-0.6	0.6

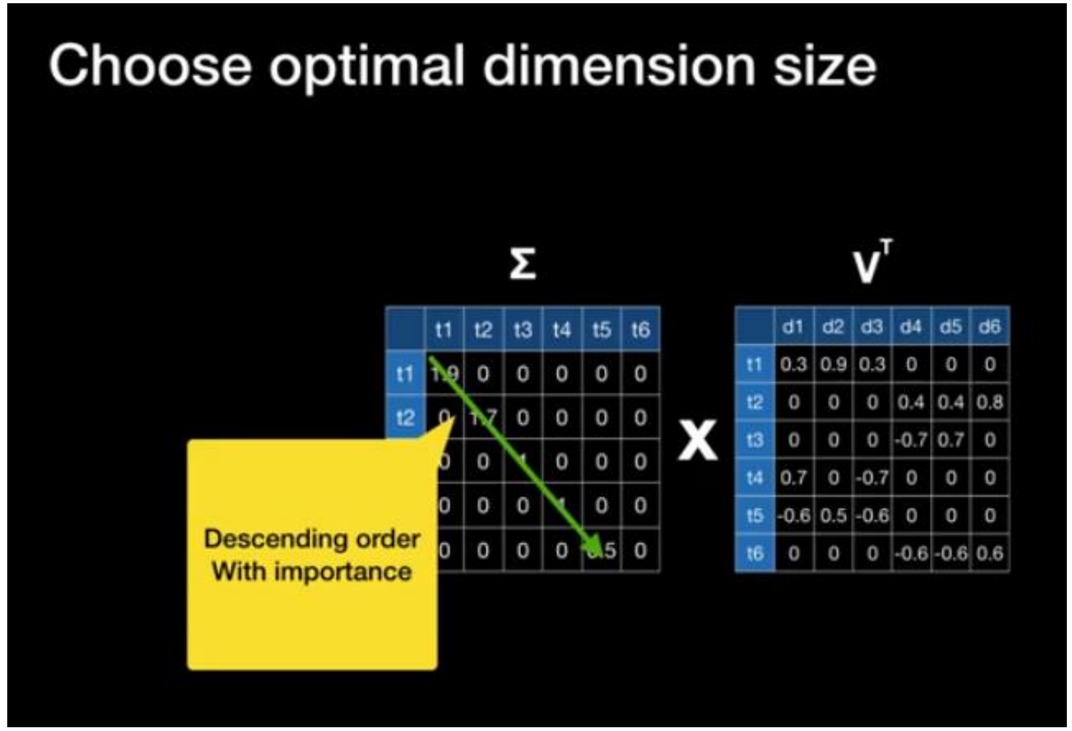
Document matrix for topic

잠재의미분석(p. 137~142)

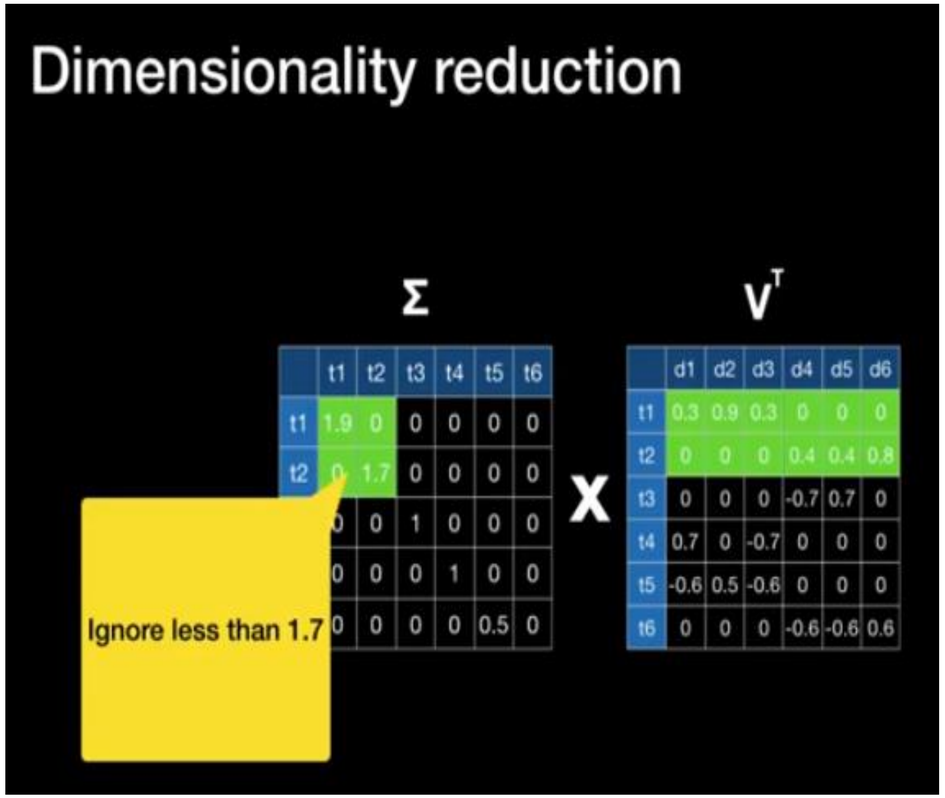


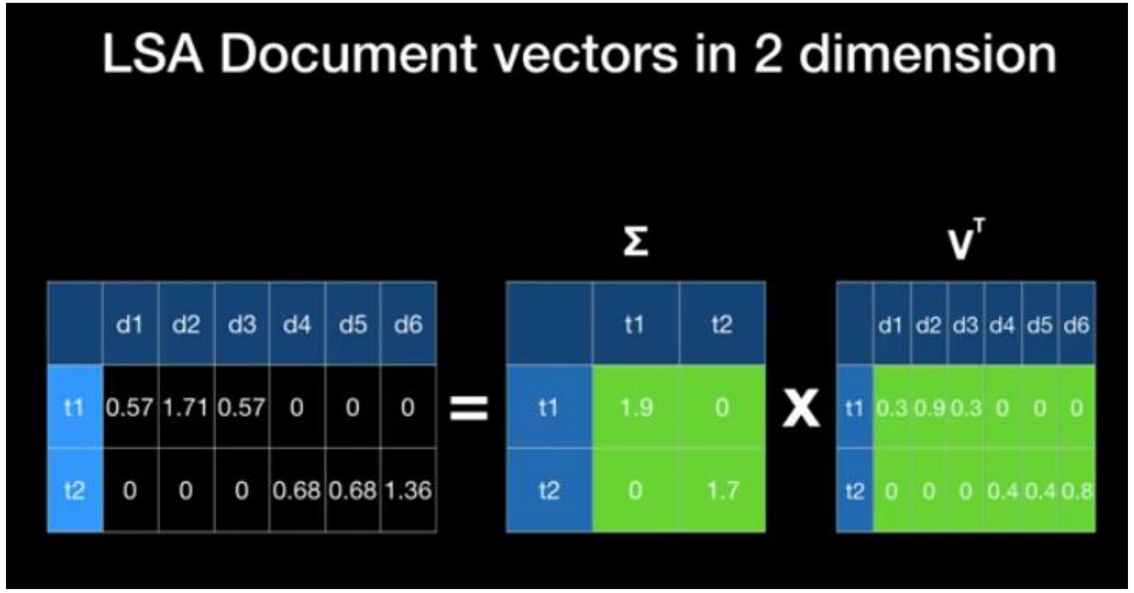
	t1	t2	t3	t4	t5	t6
t1	1.9	0	0	0	0	0
t2	0	1.7	0	0	0	0
t3	0	0	1	0	0	0
t4	0	0	0	1	0	0
t5	0	0	0	0	0.5	0

	d1	d2	d3	d4	d5	d6
t1	0.3	0.9	0.3	0	0	0
t2	0	0	0	0.4	0.4	0.8
t3	0	0	0	-0.7	0.7	0
t4	0.7	0	-0.7	0	0	0
t5	-0.6	0.5	-0.6	0	0	0
t6	0	0	0	-0.6	-0.6	0.6



잠재의미분석(p. 137~142)



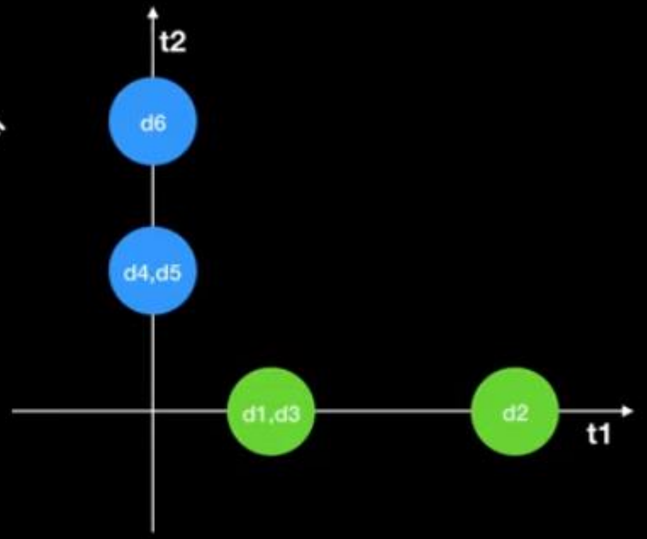


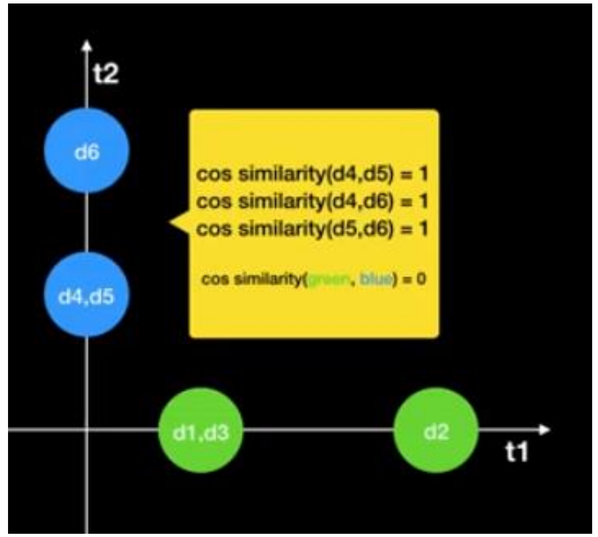
잠재의미분석(p. 137~142)

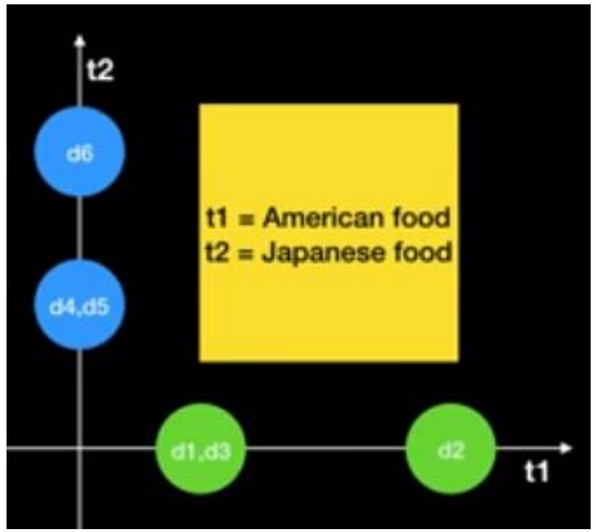
Topic Similarity

	d1	d2	d3	d4	d5	d6
t1	0.57	1.71	0.57	0	0	0
t2	0	0	0	0.68	0.68	1.36

pizza
pizza hamburger cookie
hamburger
ramen
sushi
ramen sushi







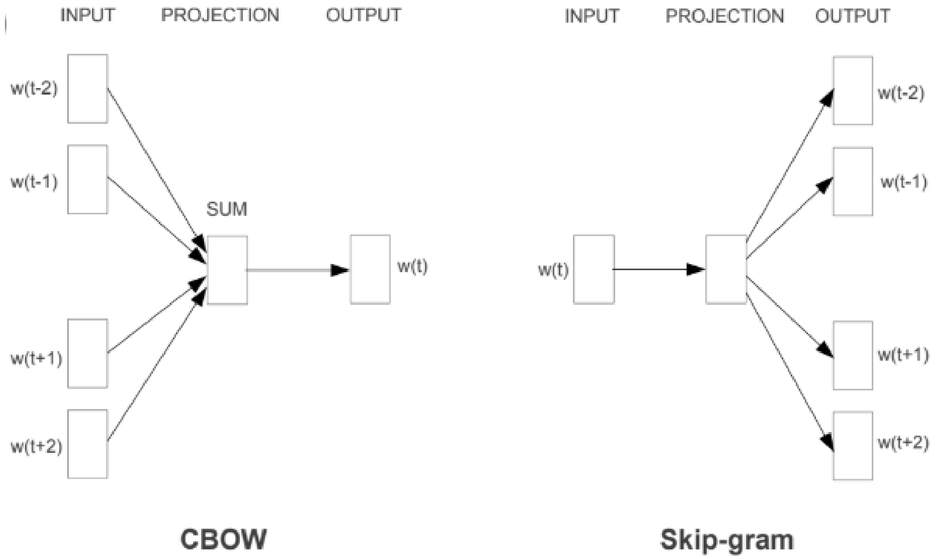

BOAZ
Thank You

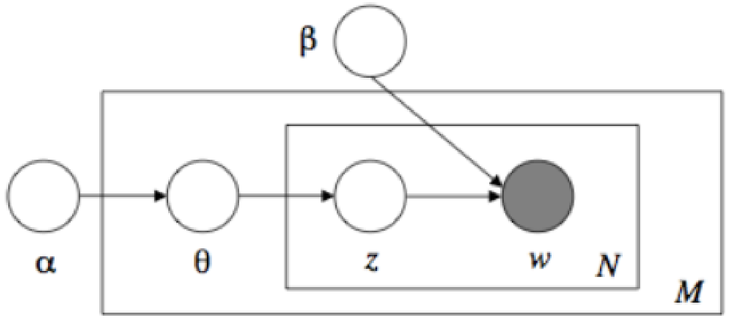
Glove (p. 143~146)

@ LDA와 Word2Vec의 단점을 극복한 Glove

Word2Vec : 말뭉치 전체 통계정보 x

LDA : 단어 간 유사도 파악 x





GloVe : **단어 간 유사도 챙기면서도 말뭉치 전체 정보 담아보자!**

Glove (p. 143~146)

@ LDA와 Word2Vec의 단점을 극복한 Glove

$$J = \sum_{i,j=1}^{V} f\left(X_{ij}\right)\left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij}\right)^2$$

목적함수를 최소화하기!

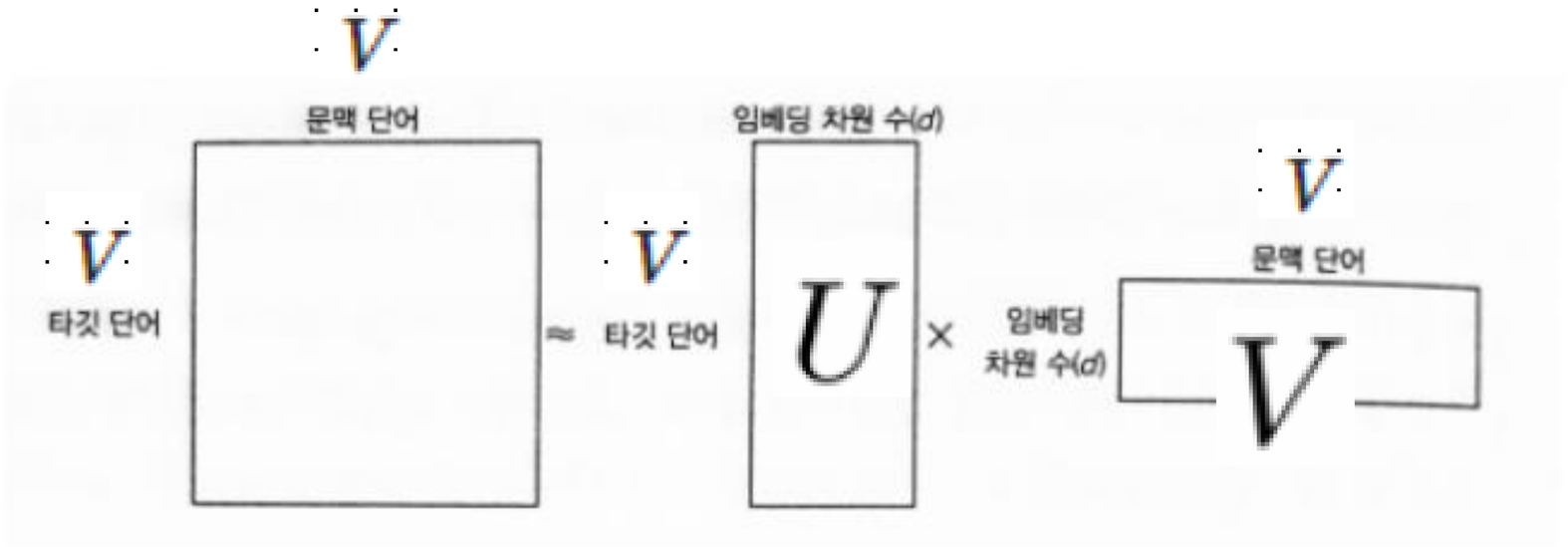
X_{ij} : 두 단어 동시 등장빈도 값

b_i b_j : Bias

w_i w_j : 단어 i, j에 해당하는 벡터

Glove (p. 143~146)

@ LDA와 Word2Vec의 단점을 극복한 Glove



U,V 랜덤으로 초기화, 목적함수를 최소화하는 방향으로 그래디언트 디센트방법 업데이트

U를 임베딩으로 사용가능

U,V 이거붙이거나 더해서 임베딩으로 사용 가능

Swivel (p. 147~150)

@ PMI의 단점을 극복한 Swivel

두 가지 경우 1. 말뭉치에 동시 등장한 케이스가 한 건이라도 있는 경우

2. 말뭉치에 동시 등장한 케이스가 한 건도 없는 경우

$x_{ij} > 0$ 일때, $\frac{1}{2} f(x_{ij})(w_i^\top \tilde{w}_j - \mathbf{pmi}(i;j))^2$

f가 클수록 벡터 간 내적값이 PMI값과 비슷해야 학습손실 줄어들어

=동시 등장 많아질수록

Swivel (p. 147~150)

@ PMI의 단점을 극복한 Swivel

$$pmi(i,j) = \log \frac{P(i,j)}{P(i)P(j)}$$

(, $x_{i*} = \sum_j x_{ij}$, $|D| = \sum_{i,j} x_{ij}$)

$$= \log \frac{x_{ij}|D|}{x_{i*}x_{*j}} = \log x_i + \log |D| - \log x_{i*} - \log x_{*j}$$

두 단어가 한 번도 등장하지 않았을 때 PMI는 음의 무한대로 발산

Swivel (p. 147~150)

@ PMI의 단점을 극복한 Swivel

두 가지 경우 1. 말뭉치에 동시 등장한 케이스가 한 건이라도 있는 경우

2. 말뭉치에 동시 등장한 케이스가 한 건도 없는 경우

$x_{ij} = 0$ 일때,

$$\frac{1}{2} f(x_{ij})(w_i^\top \tilde{w}_j - \mathbf{pmi}(i;j))^2$$

$$pmi(i,j) = \log \frac{x_{ij}|D|}{x_{i*}x_{*j}} = \log x_i + \log |D| - \log x_{i*} - \log x_{*j}$$

따라서 $\log \left[1 + \exp\left(w_i^\top \tilde{w}_j - \mathbf{pmi}^*(i;j)\right)\right]$

Swivel (p. 147~150)

@ PMI의 단점을 극복한 Swivel

두 가지 경우 1. 말뭉치에 동시 등장한 케이스가 한 건이라도 있는 경우

2. 말뭉치에 동시 등장한 케이스가 한 건도 없는 경우

$x_{ij} = 0$ 일때, $\log\left[1 + \exp\left(w_i^\top \tilde{w}_j - \mathbf{pmi}^*(i;j)\right)\right]$

고빈도 단어인데 등장 빈도가 0 -> $\log x_{*j}$ $\log x_{i*}$ 감소

-> 내적 값 작게 만들어야 학습 손실 감소

어떤 임베딩을 사용할 것인가? (p. 151~159)

@단어 수준 임베딩

Word2Vec

FastText

GloVe

Swivel

어떤 임베딩을 사용할 것인가? (p. 151~159)

@단어 수준 임베딩

Word2Vec

FastText

GloVe

Swivel

- 통사론적, 의미론적 관계가 임베딩에 얼마나 잘 녹아 있는지 평가해보자!

  ㄴ 유사도 평가 / 단어 유추 평가 / 시각화

어떤 임베딩을 사용할 것인가? (p. 151~159)

@유사도 평가

- 일련의 단어 쌍을 미리 구성한 후, 사람이 평가한 점수와 단어 벡터 간 코사인 유사도 사이의 상관관계 계산

Word 1	Word 2	Score
Computer	News	4.47
Tiger	Cat	8.00
Mars	Water	2.94

어떤 임베딩을 사용할 것인가? (p. 151~159)

@유사도 평가

- 학습이 완료된 임베딩을 평가해보니…

어떤 임베딩을 사용할 것인가? (p. 151~159)

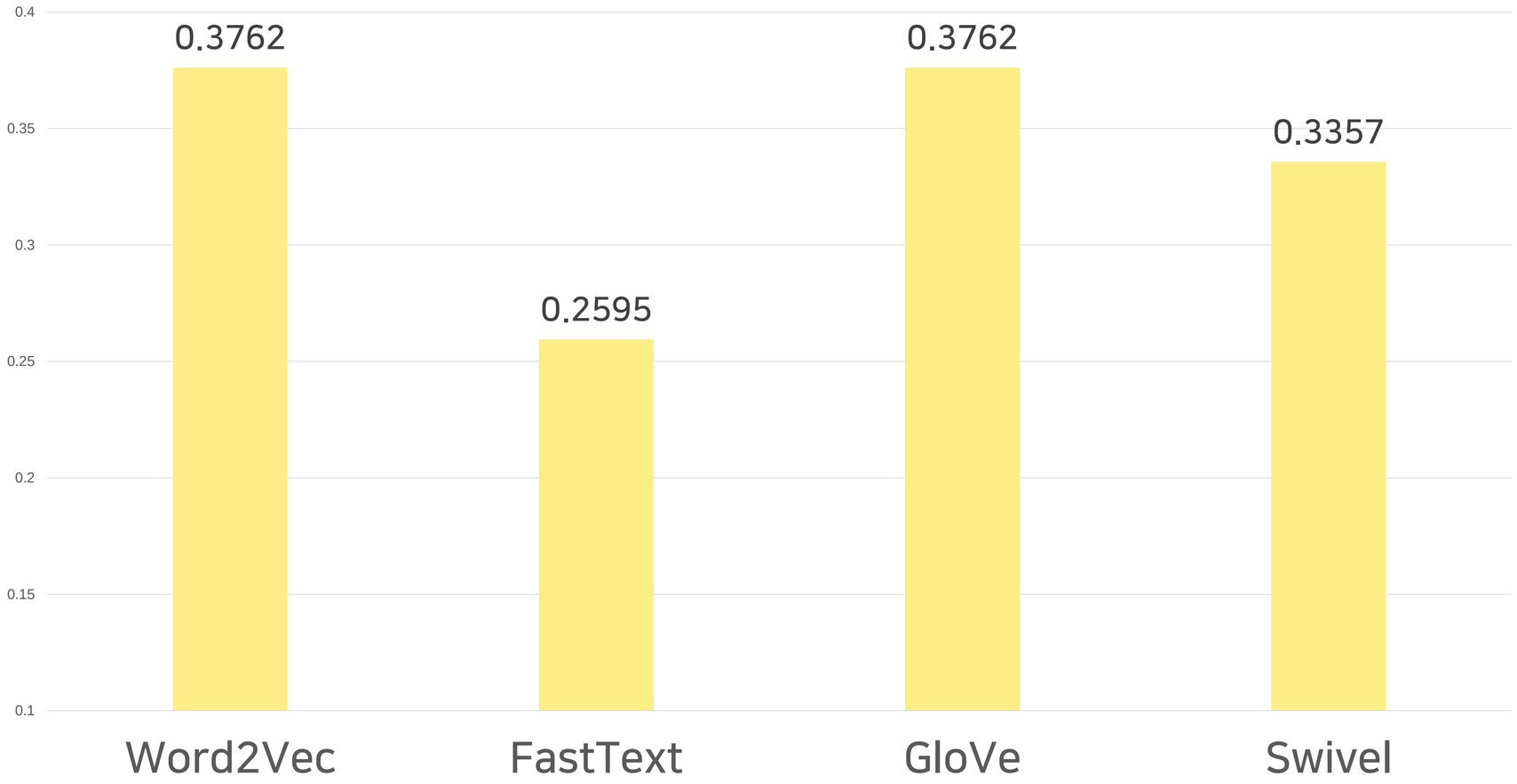
@단어 유추 평가

- 단어 벡터 간 계산을 통해 (갑-을+병)이라는 질의에 '정'을 도출해낼 수 있는지 평가
  ㄴ ex. (한국 – 서울 + 도쿄) 라는 질의에 '일본'을 도출해낼 수 있는지 평가

- (갑-을+병)에 해당하는 벡터에 대해 코사인 유사도가 가장 높은 벡터에 해당하는 단어가 '정'인지 확인

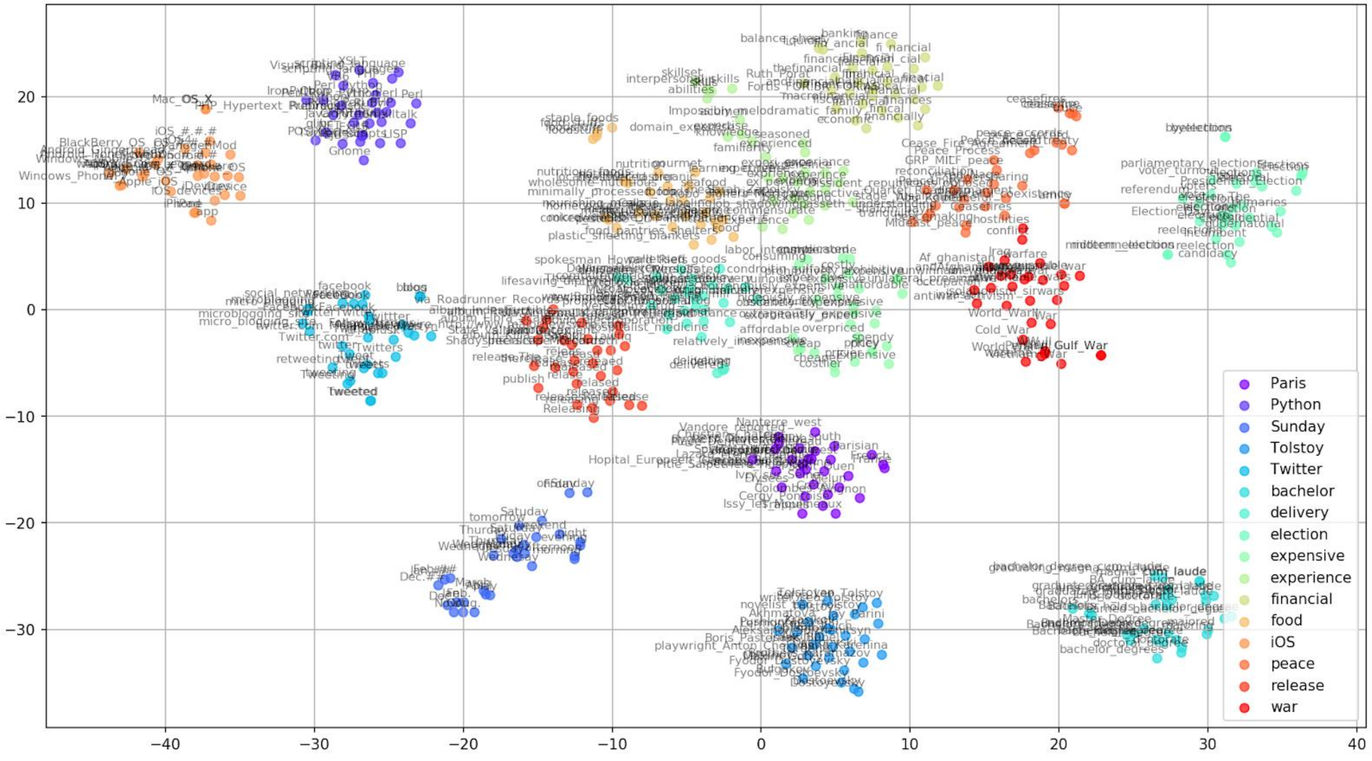
어떤 임베딩을 사용할 것인가? (p. 151~159)

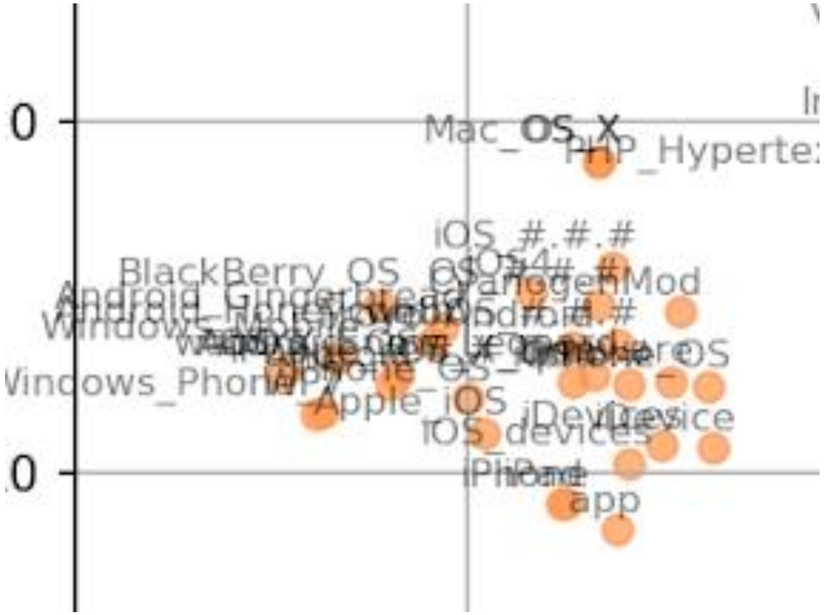
@단어 유추 평가

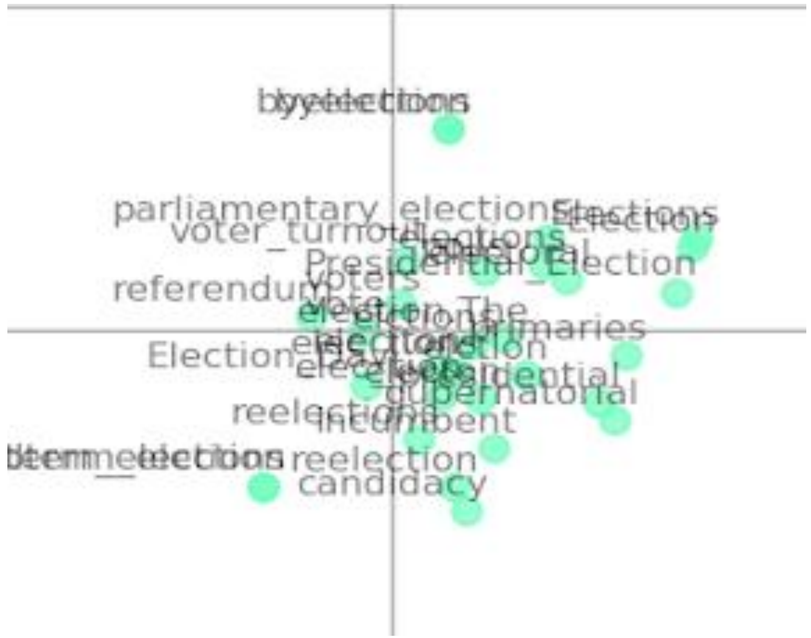


어떤 임베딩을 사용할 것인가? (p. 151~159)

@단어 임베딩 시각화 _ t-SNE







가중 임베딩 (p. 160~171)

@가중 임베딩

: 단어 임베딩을 문장 수준 임베딩으로 확장하는 방법 by. 프린스턴 대학교 연구팀 (Arora et al., 2016)

- 가정: 문서 내 단어 등장은 저자가 생각한 주제에 의존
(= 주제에 따라 단어 사용 양상 달라짐)

> 주제 벡터 개념 도입

가중 임베딩 (p. 160~171)

@주제 벡터 c_s 가 주어졌을 때 어떤 단어 w가 나타날 확률

$$P(W|c_s) = \alpha P(w) + (1-\alpha)\frac{exp(\widetilde{c_s} * v_w)}{Z}$$

$P(w)$: 주제와 상관 없이 단어 w가 등장할 확률

$\frac{exp(\widetilde{c_s} * v_w)}{Z}$: 단어 w가 주제와 관련을 가질 확률 (주제 벡터' $\widetilde{c_s}$ 와 v_w가 유사할 수록 값 ↑)

가중 임베딩 (p. 160~171)

@문장 등장 확률

- 단어 시퀀스는 곧 문장.
  > 문장 등장 확률은 문장에 속한 모든 단어들이 등장할 확률의 누적 곱

$$\boldsymbol{P(s|c_s)} \propto \sum_{w \in S} \log_P(w|c_s) = \sum_{w \in S} f_w(\tilde{c}_s)$$

$$\sum_{w \in S} f_w(\tilde{c}_s) = \text{constant} + \frac{\frac{(1-\alpha)}{\alpha Z}}{P(W)+(1-\alpha)/\alpha Z} \tilde{c}_s * v_w$$

가중 임베딩 (p. 160~171)

@문장 등장 확률

$$\text{argmax } f_w(\tilde{c}_s) = \text{argmax } [\text{constant} + \frac{a}{P(W)+a} \tilde{c}_s * v_w]$$

… 정리

$$\text{argmax} \sum_{w \in s} f_w(\tilde{c}_s) \propto \sum_{w \in s} \frac{a}{P(\omega)+a} v_w$$

가중 임베딩 (p. 160~171)

@가중 임베딩 성능 평가

- 학습) 문장(영화 댓글) + 레이블(긍정/부정)
- 테스트) 테스트 문장이 들어오면 벡터의 합으로 만들고, 코사인 유사도가 가장 높은 학습 데이터 문장 찾아 레이블 리턴

잠재의미 분석 (p.177~182)

단어-문서, TF-IDF, 단어-문맥, PMI 행렬에 특이값 분해로 차원축소 시행

단어에 해당하는 벡터를 취해 임베딩

Input

행: 문서　　열: 단어

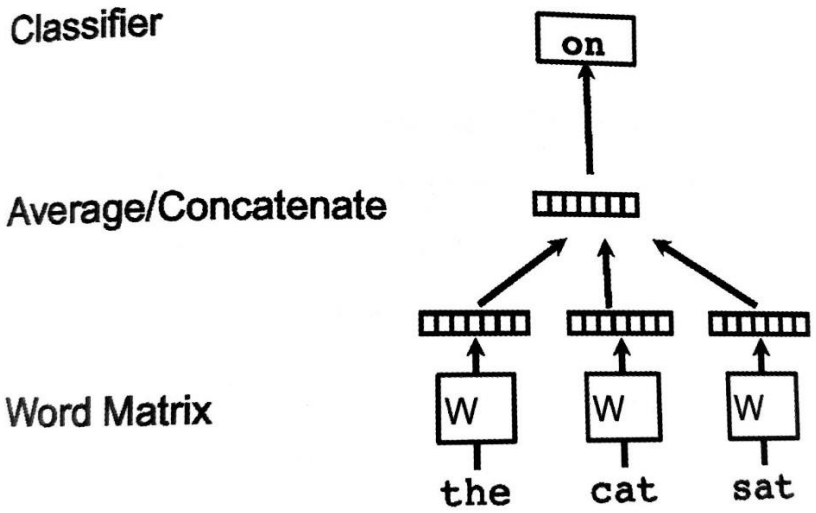
원소가 0인 단어, 즉 정보성이 낮은 대부분의 단어는 무시

열을 줄이는 선형변환을 통해 코사인 유사도가 가장 높은 단어들을 뽑는다

Doc2Vec (p.183~190)

Word2Vec에 이어 구글 연구 팀이 개발한 **문서 임베딩** 기법

Le&Mikelov(2014) : 이전 **단어 시퀀스 k개**가 주어졌을 때 그 다음 단어를 맞추는 언어 모델 개발



<The cat sat on the mat 문장 예시, k = 3>

Doc2Vec (p.183~190)

$$L = \frac{1}{T}\Sigma \sum_{t=k}^{T-k} \log p(w_t | w_{t-k}, \ldots, w_{t-1})$$

w_t: 문장의 t번째 단어

y_i: 말뭉치 전체 어휘 집합 중 i번째 단어에 해당하는 점수

학습 데이터 문장 하나의 단어 개수가 T일 때 해당 문장의 로그 확률의 평균

이 값을 최대화 하는 과정에서 학습을 진행

Doc2Vec 언어 모델 스코어 (p.183~190)

$$p(w_t|w_{t-k},\ldots,w_{t-1}) = \exp(y_{w_t})/\sum_i \exp(y_i)$$

$$Y = b + U \times h(w_{t-k},\ldots,w_{t-1};W)$$

y_i: 말뭉치 전체 어휘 집합 중 i번째 단어에 해당하는 점수

U: 어휘 집합 크기 x 임베딩 차원 수

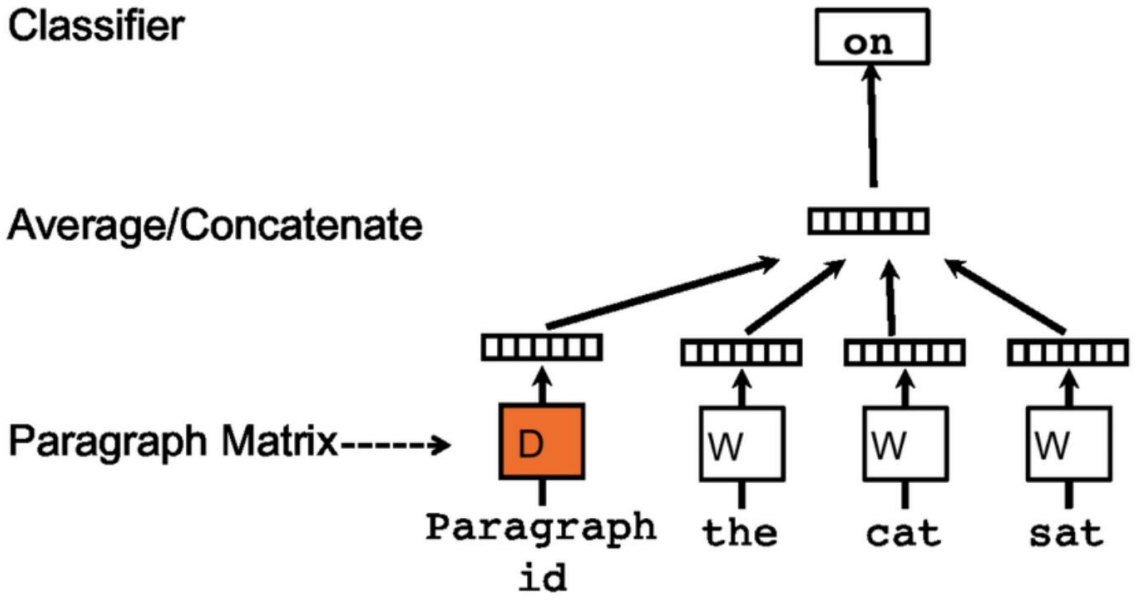
이전 k개 단어들을 W라는 단어 행렬에서 참조한 뒤 평균을 취하거나 이어붙임.

U라는 행렬을 내적하고, bias를 더해준 뒤 소프트 맥스를 취함.

다음 단어를 맞추는 과정에서 W행렬의 벡터들이 업데이트

Doc2Vec 언어 모델 스코어 (p.183~190)

PV-DM (Le&Mikelov,2014)



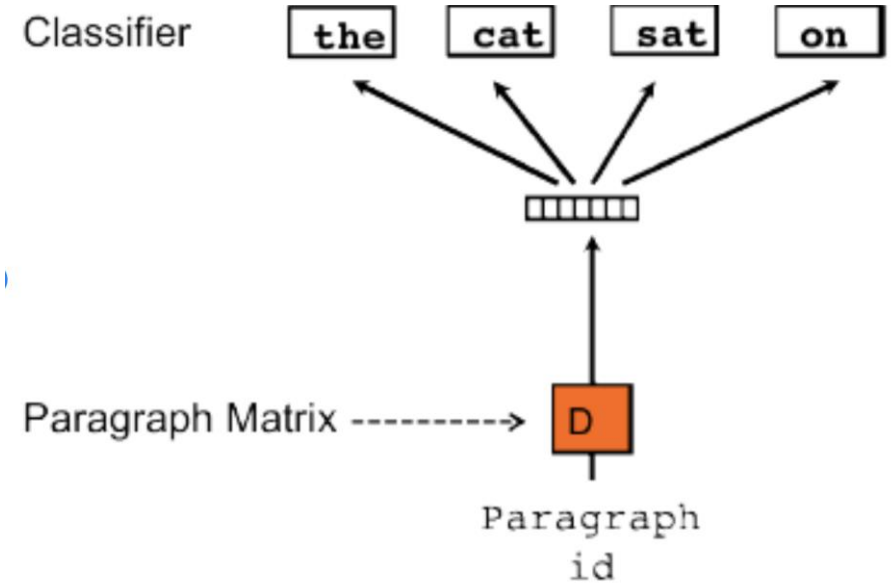
이전 K개 단어들과 문서 ID를 넣어서 다음단어를 예측

해당 문서의 주제 정보를 함축.

문서 임베딩은 동일한 문서 내 존재하는 모든 단어와 함께 학습될 기회를 갖는다.

Doc2Vec 언어 모델 스코어 (p.183~190)

PV-DBOW (Le&Mikelov,2014)

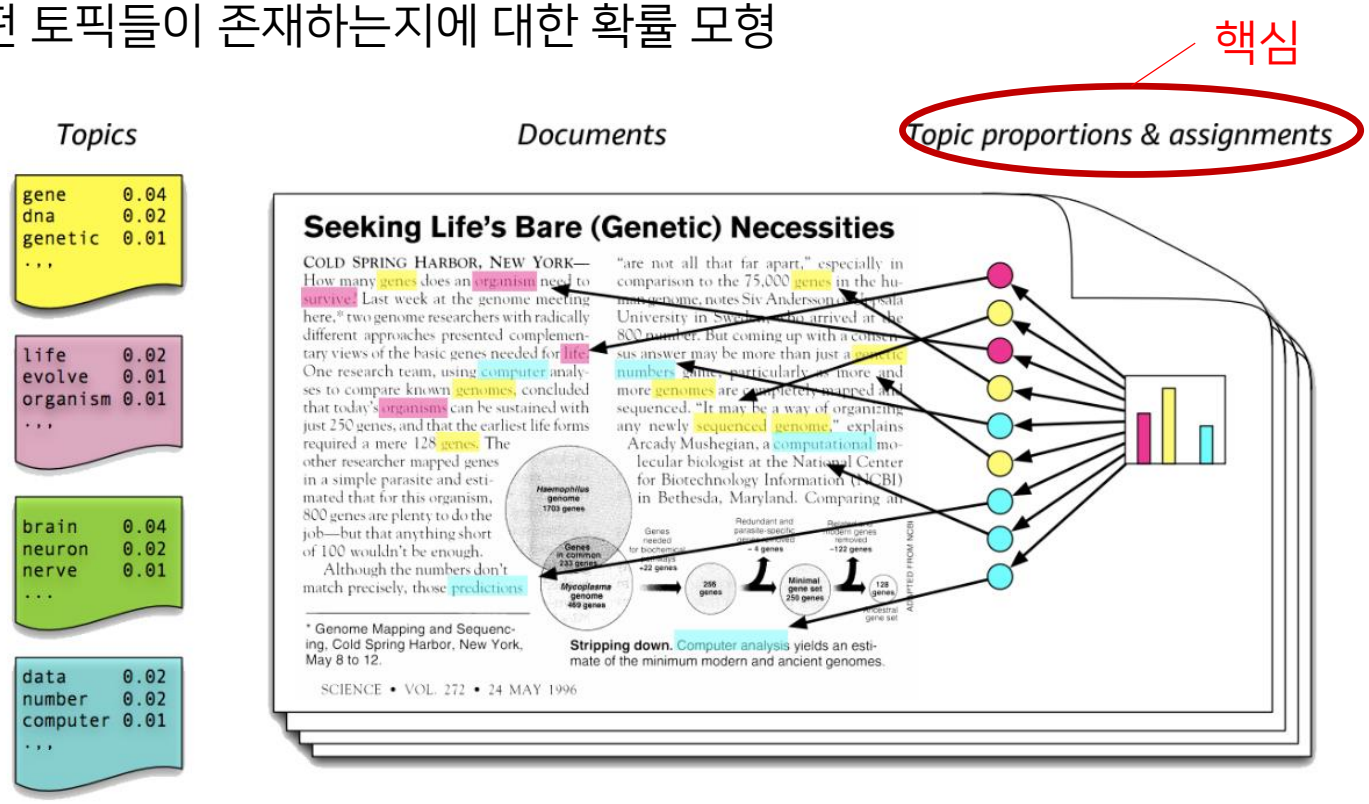


문서 ID를 가지고 문맥 단어를 맞춘다.

문서 ID에 해당하는 문서 임베딩엔, 문서에 등장하는 모든 단어의 의미정보가 반영

잠재 디리클레 할당 (p.190~199)

각 문서에 어떤 토픽들이 존재하는지에 대한 확률 모형

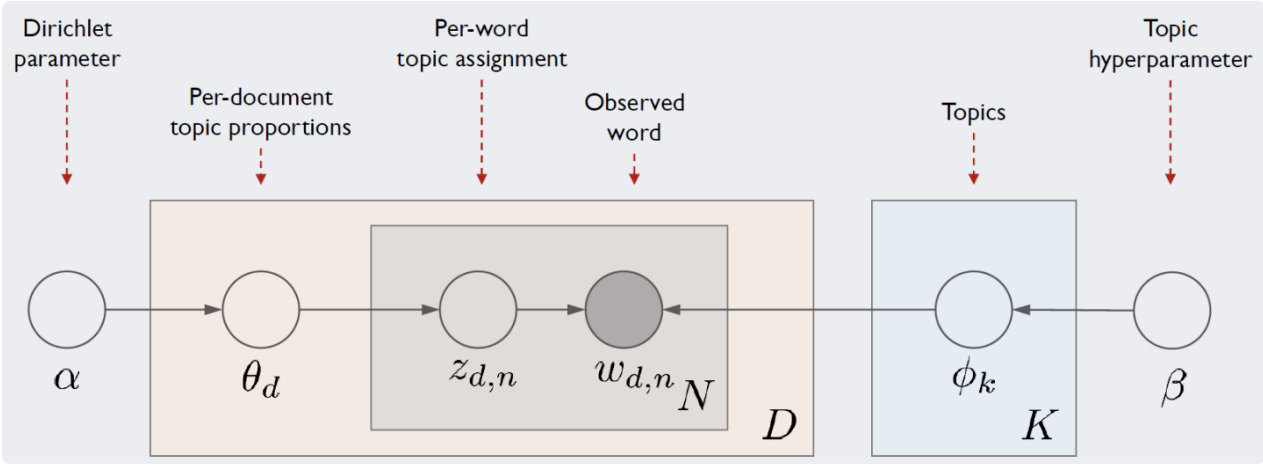


LDA는 특정 토픽에 특정 단어가 나타날 확률을 출력

말뭉치로부터 얻은 토픽 분포로부터 토픽을 뽑은 뒤, 해당 토픽에 해당하는 단어들을 뽑는다.

잠재 디리클레 할당 (p.190~199)

LDA 문서생성 과정



D: 말뭉치 전체 문서 개수

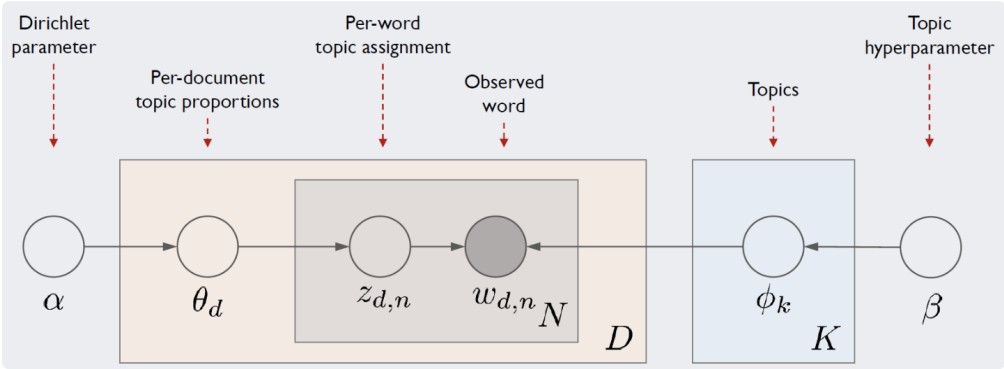
K: 전체 토픽 수

N: d번째 문서의 단어 수

$w_{d,n}$: d번째 문서에 등장한 n번째 단어 (유일하게 관찰 가능한 변수)

하이퍼 파라미터 α, β를 제외한 모든 잠재 변수를 추정해야함

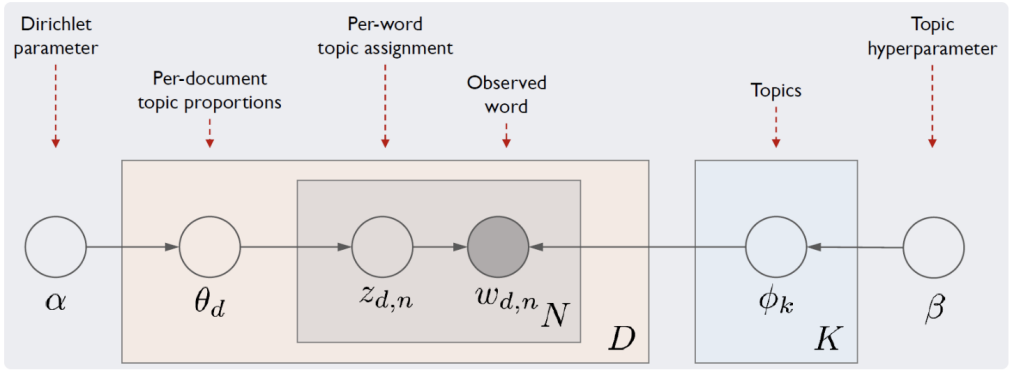
잠재 디리클레 할당 (p.190~199)



$\emptyset_k$: k번째 토픽에 해당하는 벡터. 각 요소 값은 해당 단어가 k번째 토픽에서 차지하는 비중을 나타냄.

단어	topic-1	topic-2	topic-3
코로나 바이러스	0.000	0.393	0.000
우한폐렴	0.000	0.313	0.000
AWS	0.119	0.000	0.000
데이터 엔지니어링	0.181	0.000	0.000
Hadoop	0.276	0.000	0.000
Spark	0.142	0.000	0.000

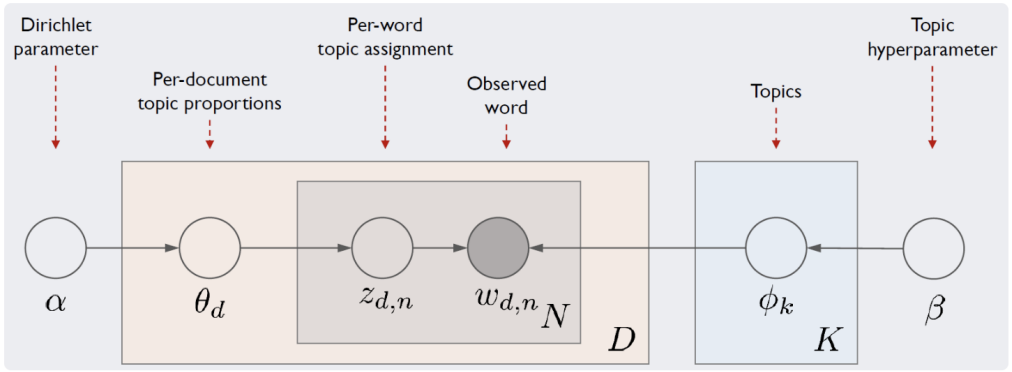
잠재 디리클레 할당 (p.190~199)



θ_d: d번째 문서가 가진 토픽 비중을 나타내는 벡터

문서	topic-1	topic-2	topic-3
문서1	0.400	0.000	0.600
문서2	0.000	0.600	0.400
문서3	0.375	0.625	0.000

잠재 디리클레 할당 (p.190~199)



$z_{d,n}$: d번째 문서 n번째 단어가 어떤 토픽인지를 나타내는 변수

$w_{d,n}$: d번째 문서 내에 n번째로 등장하는 단어

LDA는 토픽의 단어 분포와 문서의 토픽 분포의 결합으로 문서 내 단어들이 생성된다고 가정

실제로 관찰 가능한 말뭉치를 가지고 토픽의 단어 분포, 문서의 토픽 분포를 추정

토픽의 단어 분포와 토픽 분포의 결합 확률이 커지도록!!!

잠재 디리클레 할당 (p.190~199)

LDA 단어 생성 과정

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

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

$p(z, \emptyset, \theta|w)$를 최대로 만드는 $z, \emptyset, \theta$를 찾아야한다.

사후확률 분포 $-$ $p(z, \emptyset, \theta|w) = p(z, \emptyset, \theta, w)/p(w)$

$p(w)$는 베이지안 확률 모델에서 증거에 해당, 잠재변수의 모든 경우의 수를 고려한 각 단어의 등장확률 구하기 어렵게 때문에 깁스 샘플링을 사용해 사후 확률을 근사

잠재 디리클레 할당 (p.190~199)

깁스 샘플링 예시

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

단어 5개로 구성된 문서1의 모든 단어에 주제가 이미 할당되어 있다고 가정
같은 단어라도 토픽이 다를 수 있기 때문에, 각 단어별로 토픽 분포 생성

단어	topic-1	topic-2	topic-3
천주교	1	0	35
시장	50	0	1
가격	42	1	0
불교	0	0	20
무역	10	8	1
...	...	...	...

잠재 디리클레 할당 (p.190~199)

깁스 샘플링 예시

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

문서1의 단어별 topic 분포

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

word-topic 행렬

단어	topic-1	topic-2	topic-3
천주교	1	0	35
시장	50	0	1
가격	42	1	0
불교	0	0	20
무역	10	7 = (8 - 1)	1
...	...	...	...