

NLP 스터디 WEEK4

CH5. 문장 수준 임베딩

5.1~5.4

NLP 스터디  
정지용

# 목차

- 잠재 의미 분석 (LSA)
- Doc2Vec
- 잠재 디리클레 할당 (LDA)
- ELMo

# 잠재 의미 분석 (LSA)

- 행렬 분해 (Matrix Factorization) 기반 임베딩 방법
- 단어 - 문서로 이루어진 행렬 (ex. TF-IDF 행렬)을 SVD를 통해 문서 벡터 생성

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

	문서1	문서2	문서3	...
김밥	0.53	0.0	0.22	...
라면	0.0	0.48	0.11	...
돈까스	0.58	0.44	0.02	...
햄버거	0.87	0.14	0.0	...
...	...	...	...	...



	d 차원의 벡터
김밥	[0.24, 0.32, ...]
라면	[0.11, 0.42, ...]
돈까스	[0.22, 0.45, ...]
햄버거	[0.62, 0.17, ...]
...	...

$$\times \begin{bmatrix} \sigma_1 & 0 & 0 & 0 \\ 0 & \sigma_2 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \sigma_d \end{bmatrix} \times$$

$T$

	d 차원의 벡터
문서1	[0.24, 0.32, ...]
문서2	[0.11, 0.42, ...]
문서	[0.22, 0.45, ...]
...	...

# Doc2Vec

- Word2Vec에 이어 구글 연구 팀이 개발한 문서 임베딩 방법
- 기본 모델

- 이전  $k$  개의 단어를 보고 다음 단어를 맞추는 모델
- 단어  $T$ 개가 주어졌을 때,

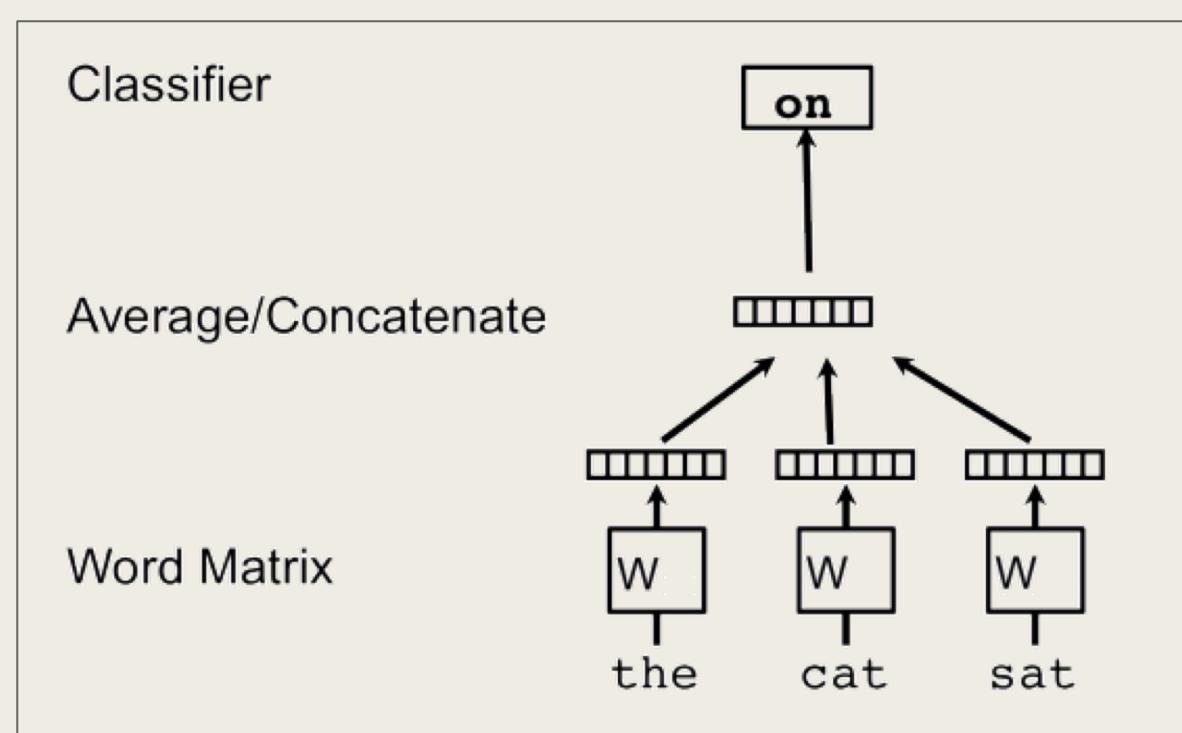
다음 로그확률 평균을 최대화 하는 방법으로 학습

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

- 각 확률값은 softmax로 계산

$$p(w_t | w_{t-k}, \dots, w_{t+k}) = \frac{e^{y_{w_t}}}{\sum_i e^{y_i}}$$

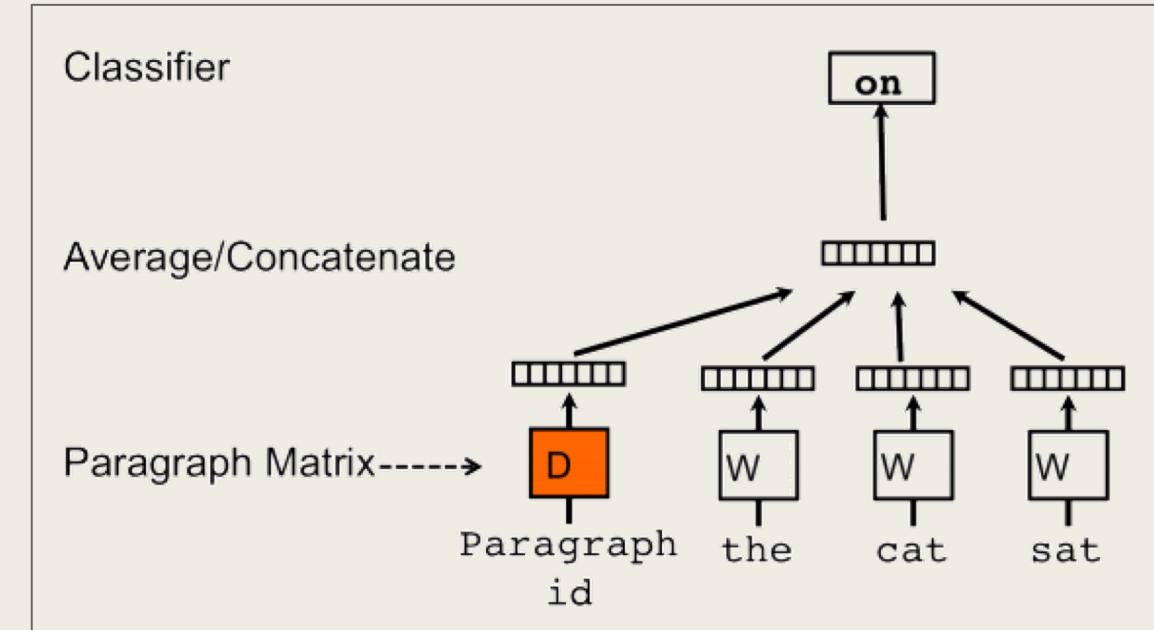
$$y = b + U h(w_{t-k}, \dots, w_{t+k}; W)$$



# Doc2Vec

## ■ PV-DM 모델

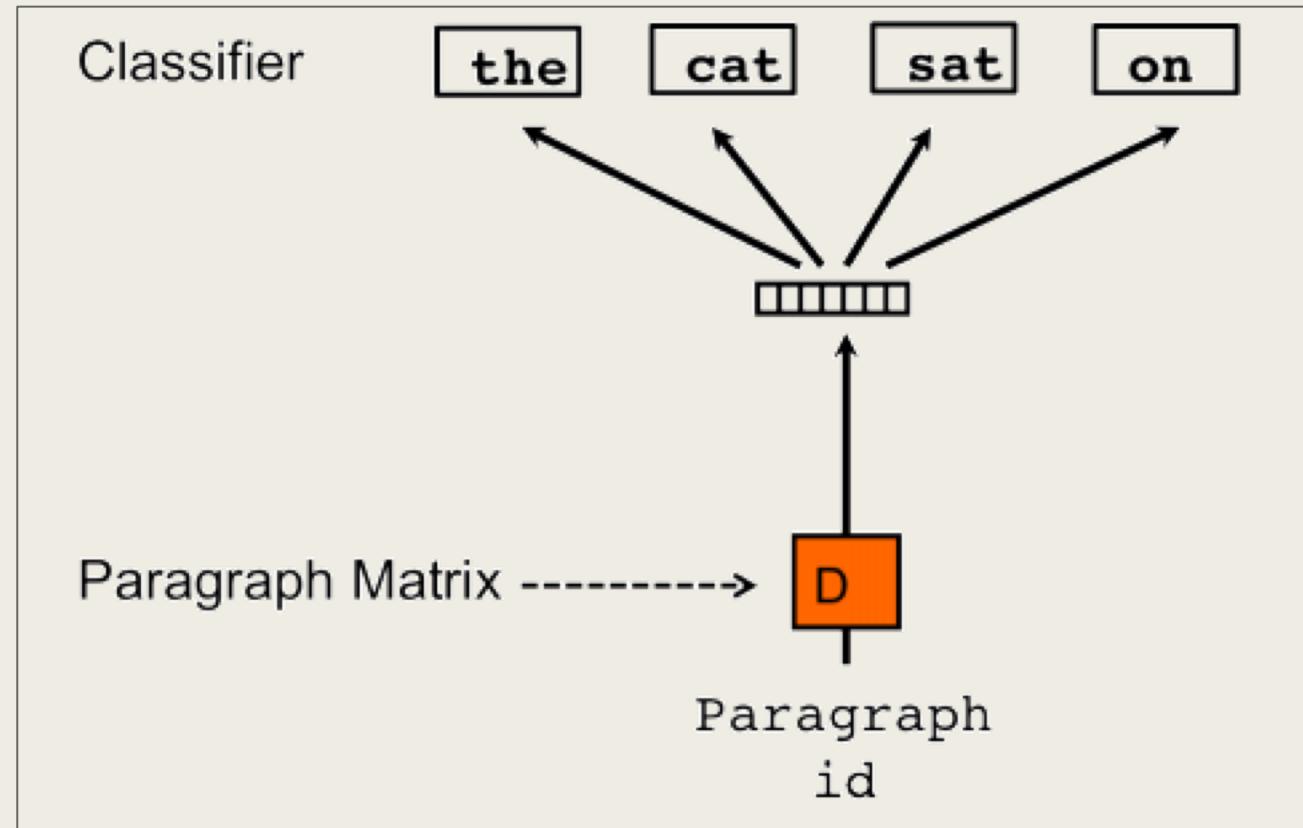
- 학습할 때, 모든 k 개의 단어와 함께 문서벡터를 참조하여  $h$  함수에 입력
- 단어와 같이 학습하기 때문에 문서 내 모든 단어 의미 포함
- 단어 등장 순서를 고려하기 때문에 백오브워즈 대비 강점



# Doc2Vec

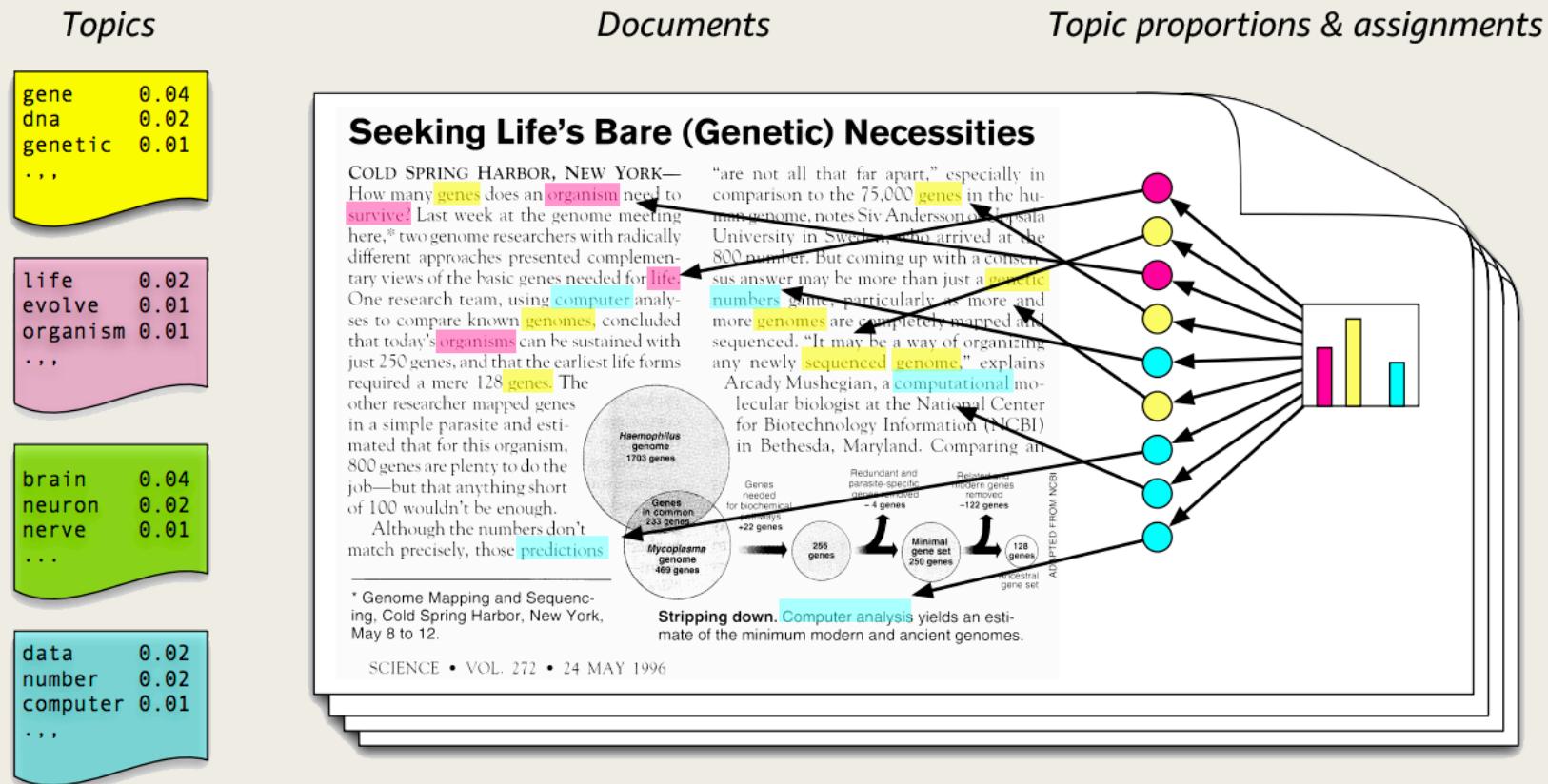
## ■ PV-DBOW 모델

- Word2Vec 의 Skip-Gram 모델을 본뜬 모델
- 문서 벡터를 토대로 문맥단어를 맞추는 방식



# 잠재 디리클레 할당 (LDA)

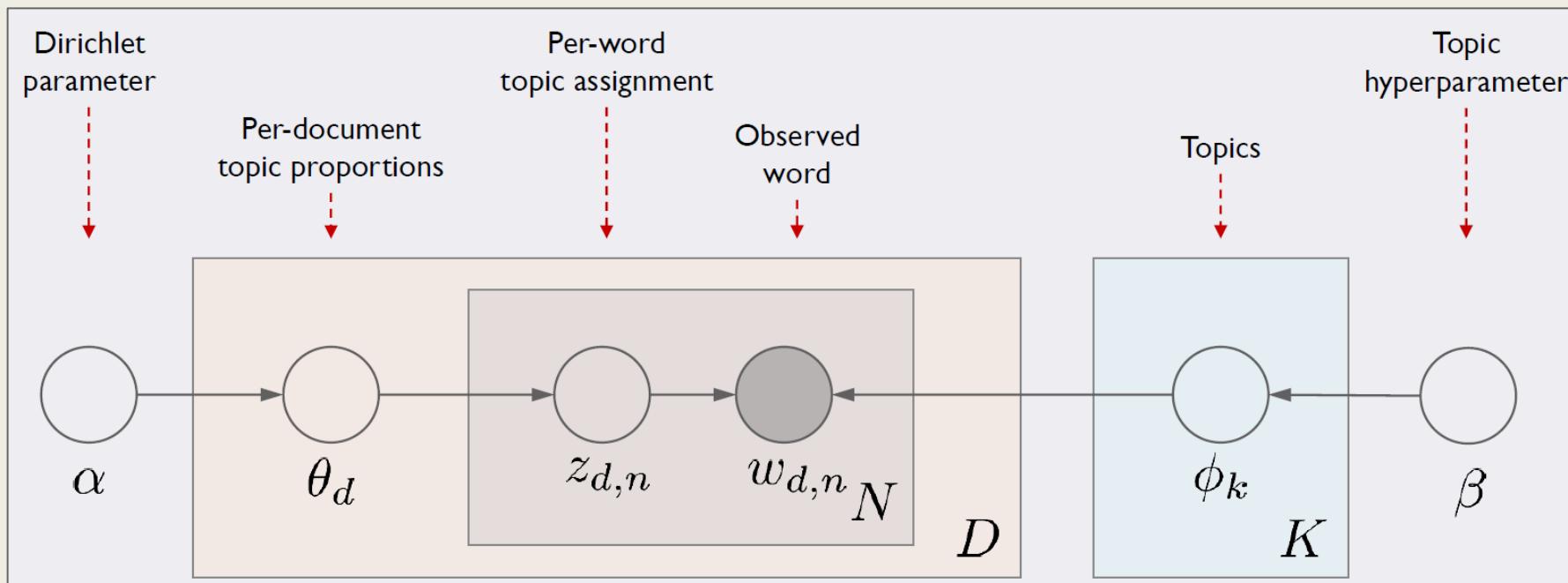
- 문서별 토픽 분포 & 토픽별 단어 분포에 대한 모형



# 잠재 디리클레 할당 (LDA)

## ■ 아키텍처

- $\alpha, \beta, K$ (토픽 수)는 하이퍼 파라미터
- $D$ : 문서 수,  $N$ : 문서의 단어 수
- $\theta_d$ : 문서의 토픽분포,  $\phi_k$ : 토픽의 단어 분포
- $Z_{d,n}$ :  $d$ 번째 문서의  $n$ 번째 단어의 토픽,  $W_{d,n}$ :  $d$ 번째 문서의  $n$ 번째 단어(유일한 관측 값)



# 잠재 디리클레 할당 (LDA)

## ■ 아키텍처

- $\alpha, \beta, K$ (토픽 수) 는 하이퍼 파라미터
- D : 문서 수, N : 문서의 단어 수
- $\theta_d$  : 문서의 토픽분포,  $\phi_k$  : 토픽의 단어 분포
- $Z_{d,n}$  : d번째 문서의 n번째 단어의 토픽,  $W_{d,n}$  : d번째 문서의 n번째 단어(유일한 관측 값)

- (1) Draw each per-corpus topic distributions  $\phi_k \sim Dir(\beta)$  for  $k \in \{1, 2, \dots, K\}$
- (2) For each document, Draw per-document topic proportions  $\theta_d \sim Dir(\alpha)$
- (3) For each document and each word, Draw per-word topic assignment  $z_{d,n} \sim Multi(\theta_d)$
- (4) For each document and each word, Draw observed word  $w_{d,n} \sim Multi(\phi_{z_{d,n}, n})$

# 잠재 디리클레 할당 (LDA)

- 단어-토픽 행렬 분포 ( $\phi_k$ )

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
Cucus	0.192	0.000	0.000
President	0.192	0.063	0.000

- 문서-토픽 행렬 ( $\theta_d$ )

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 (LDA)

## ■ LDA Inference

- LDA 에서 단어 생성 과정

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

- 하이퍼 파라미터( $\alpha, \beta$ ) 와  $w_{d,n}$  를 제외한 모든 변수가 미지수
- 사후확률 분포  $P(z, \phi, \theta | w)$  추정
- 갑스 샘플링 방법을 사용하여 사후확률 근사

# 잠재 디리클레 할당 (LDA)

## ■ 갑스 샘플링

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

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
$\alpha$	문서의 토픽 분포 생성을 위한 디리클레 분포 파라미터
$\beta$	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 (LDA)

## ■ 갑스 샘플링 예시

Doc1의 $i$ 번째 단어의 토픽 $z_{1,i}$	3	2	1	3	1
Doc1의 $n$ 번째 단어 $w_{1,n}$	Etruscan	trade	price	temple	market

구분	Topic1	Topic2	Topic3
Etruscan	1	0	35
market	50	0	1
price	42	1	0
temple	0	0	20
trade	10	8	1
...	...	...	...

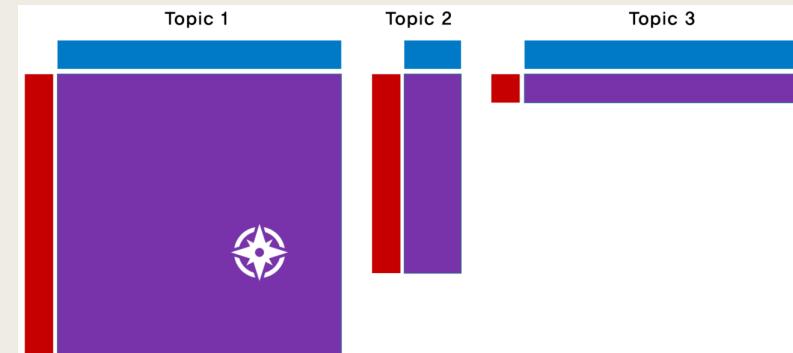


$z_{1,i}$	3	?	1	3	1
$w_{d,n}$	Etruscan	trade	price	temple	market

구분	Topic1	Topic2	Topic3
Etruscan	1	0	35
market	50	0	1
price	42	1	0
temple	0	0	20
trade	10	8-1	1
...	...	...	...

$z_{1,i}$	3	1	1	3	1
$w_{d,n}$	Etruscan	trade	price	temple	market

구분	Topic1	Topic2	Topic3
Etruscan	1	0	35
market	50	0	1
price	42	1	0
temple	0	0	20
trade	10+1	7	1
...	...	...	...



# ELMo

- Allen AI 와 미국 워싱턴 대학교 연구팀이 발표한 문장 임베딩 기법
- ELMo 이후, 자연어 처리 분야에서 모델을 **프리트레인** 한 뒤, 이를 각종 **다운스트림 태스크**에 적용하는 방식이 일반화 됨.
- **다운스트림 태스크**는 분류, 군집화 등 우리가 풀고 싶은 구체적인 문제이고 프레트레인한 모델을 다운스트림 태스크에 맞게 업데이트하는 과정을 **파인튜닝** 이라고 함.



# ELMo

- 아키텍처

- 1) 문자 단위 컨볼루션 레이어 : 각 단어의 문자들 사이의 관계 도출
- 2) 양방향 LSTM 레이어 : 단어들 사이의 관계 도출
- 3) ELMo 레이어 : 1), 2)에서 프리트레인된 출력 벡터를 가중합하여 다운스트림 태스크를 수행하는 과정에서 학습

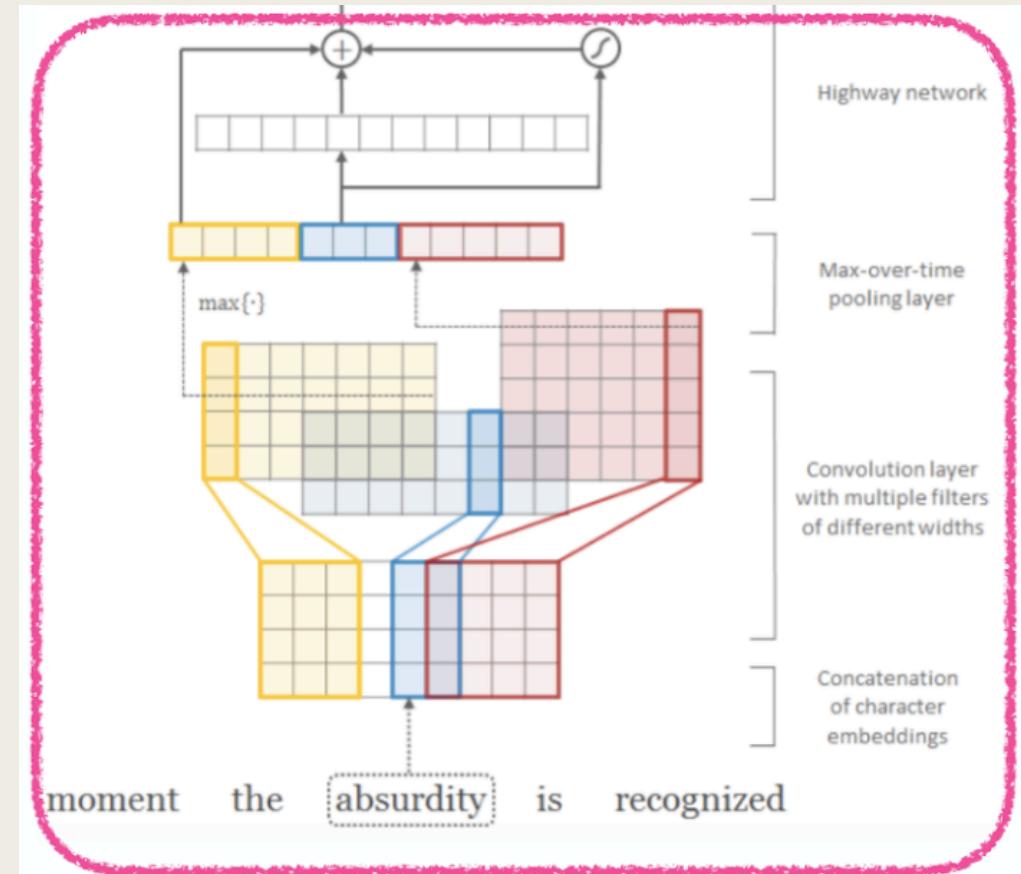
# ELMo

## ■ 문자 단위 컨볼루션 레이어

- 단어를 문자단위로 변경하여 사용 (OOV 문제 발생안함)
- 다양한 필터 사이즈로 컨볼루션 필터 생성
- 컨볼루션 필터에서 Max pooling을 통해 풀링 벡터 생성
- 풀링 벡터를 이어 붙인 후, 하이웨이 네트워크로 전송
- 하이웨이 네트워크

$$\mathbf{y} = H(\mathbf{x}, \mathbf{W}_H) \cdot T(\mathbf{x}, \mathbf{W}_T) + \mathbf{x} \cdot (1 - T(\mathbf{x}, \mathbf{W}_T))$$

- 값을 얼마나 변경( $T$ )해서 보낼지 결정
- 레이어가 많을 때, 학습이 잘 되지 않는 경우에 대한 우회 경로



# ELMo

## ■ 양방향 LSTM, 스코어 레이어

- 순방향, 역방향 LSTM 레이어를 각각 학습
- 레지듀얼 커넥션 적용
- 단어가 주어졌을 때, 그 다음 단어를 맞추는 방식으로 학습
- 최상단 레이어에서 소프트 맥스를 취하여, 정답 단어로 크로스 엔트로피 계산
- 소프트맥스 구할때는 샘플된 단어만 사용

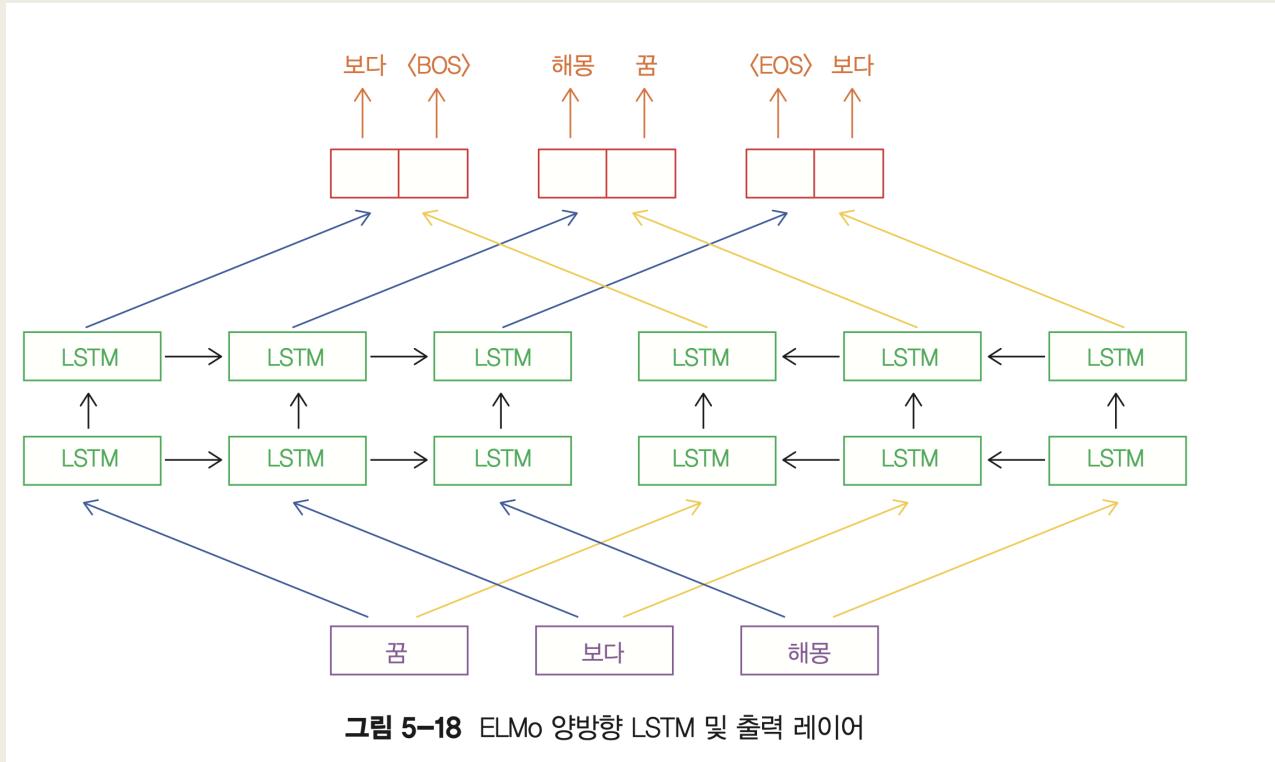


그림 5-18 ELMo 양방향 LSTM 및 출력 레이어

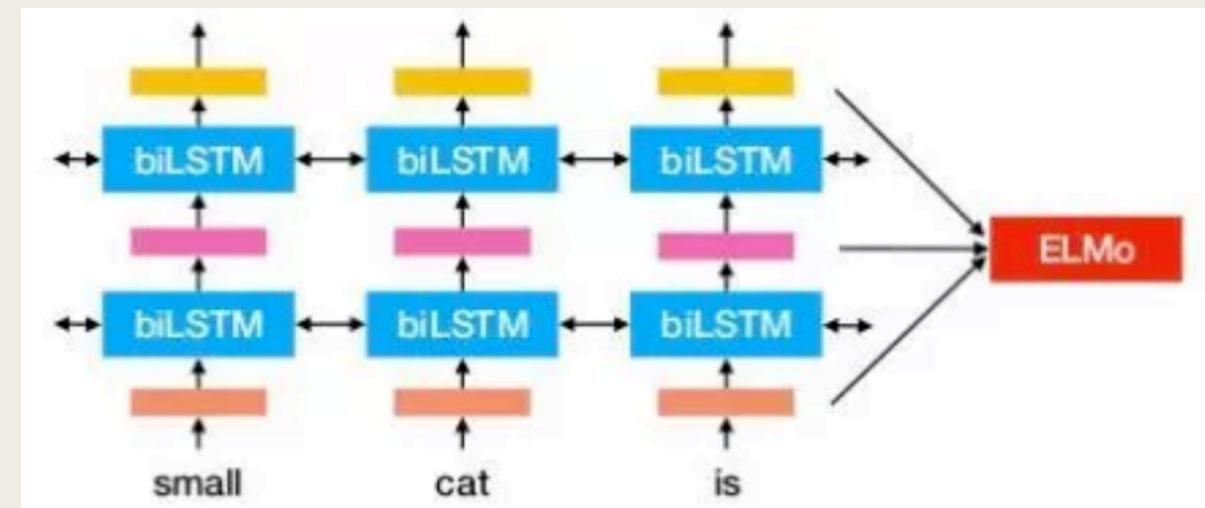
# ELMo

## ■ ELMo 레이어

- 구체적인 다운스트림 테스크를 학습하는 과정에서 임베딩 벡터 생성

$$\text{ELMo}_k^{task} = E(R_k; \Theta^{task}) = \gamma^{task} \sum_{j=0}^L s_j^{task} \mathbf{h}_{k,j}^{LM}.$$

- $\mathbf{h}_{k,j}^{LM}$  : k 번째 토큰의 j 번째 레이어의 양방향 LSTM 히든 벡터를 이어 붙인 벡터
- $s_j^{task}$  : j 번째 레이어가 해당 테스크에 얼마만큼 영향력이 있는지에 대한 가중치
- $\gamma^{task}$  : 해당 테스크에 대한 가중치



감사합니다.