ELMo-Deep contextualized word representations (2018)

FOM

Focus On Data Mining

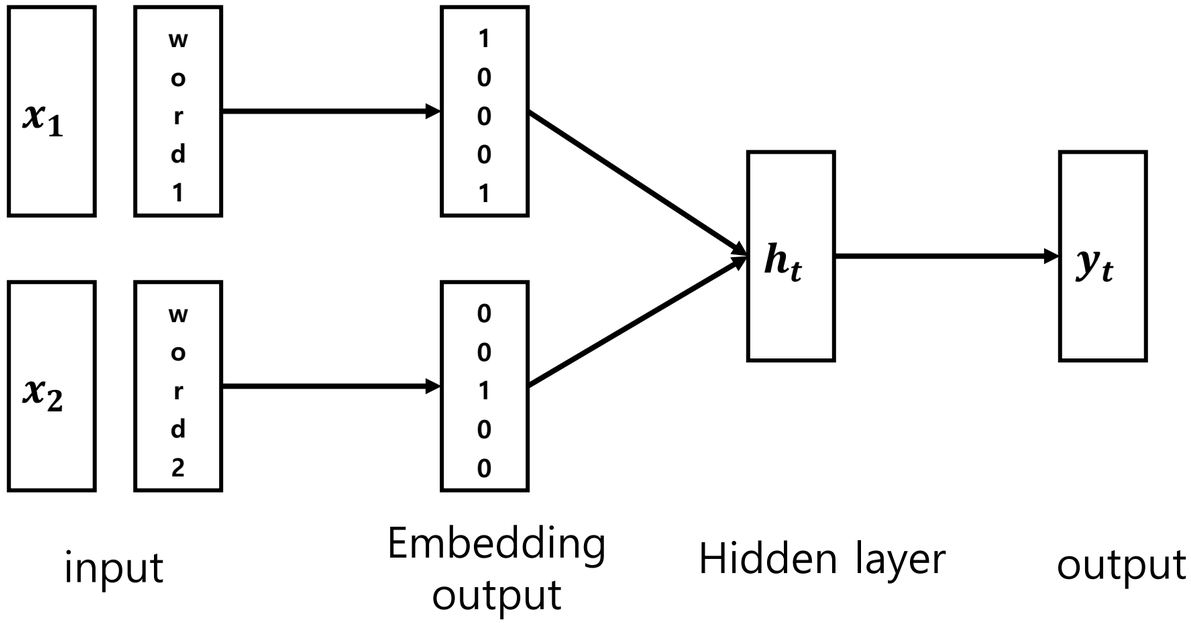
INDEX

01 Introduction

background

- Word embedding

- Word embedding이란, 자연어 데이터 (문자 데이터)를 연속형 데이터 (숫자 데이터,numeric data, 주로 문자 -> 연속형 벡터)로 변형 시켜주는 것



01 Introduction

Key point of paper and post of embedding

- 기존의 Word embedding 방법
  - sequence를 통계적 정보로 embedding하는 LSA
  - neural network를 사용해 word embedding하는 Word2vec,Glove
- 기존의 Word embedding의 문제점
  - 같은 단어라도 context에 의해 달라지는 sequence의 특징을 잡아내지 못함 (ex, present : 선물 or 현재)
    - 맥락에 따라 의미가 다른 단어들을 해결하지 못함
- Key point of paper
  - bidirectional LM (Language Model) (<- use LSTM)
  - contextualized word-embedding
  - pre-trained model

02 Related Works

Word2vec

- Word2vec

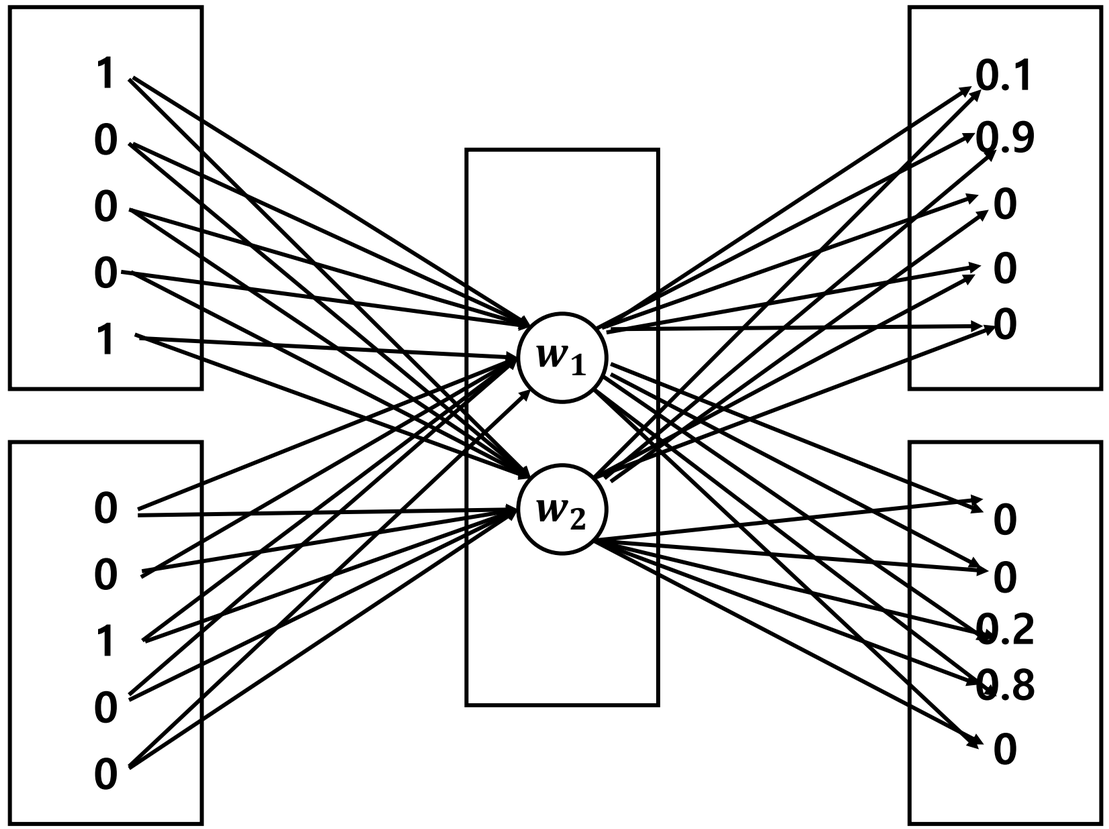
- 단순한 encoding은 word간의 similarity를 파악할 수 없음

-> embedding을 통해 similarity of neighbor words를 얻음

- 먼저 sequence를 window size를 조절하면서 word-neighbor을 찾음

->찾은 word-neighbor 조합을 one-hot encoding을 통해 변환

-> neural network를 사용해 word-neighbor의 관계를 저차원의 벡터 공간에 embedding



02 Related Works

Word2vec

- Word-neighbor

EX) “King brave man” , “queen beautiful woman”

Window size = 1

Word (input)	Neighbor (output)
King	Brave
Brave	King
Brave	Man
Man	Brave
queen	Beautiful
beautiful	Queen
Beautiful	Woman
woman	beautiful

Window size = 2

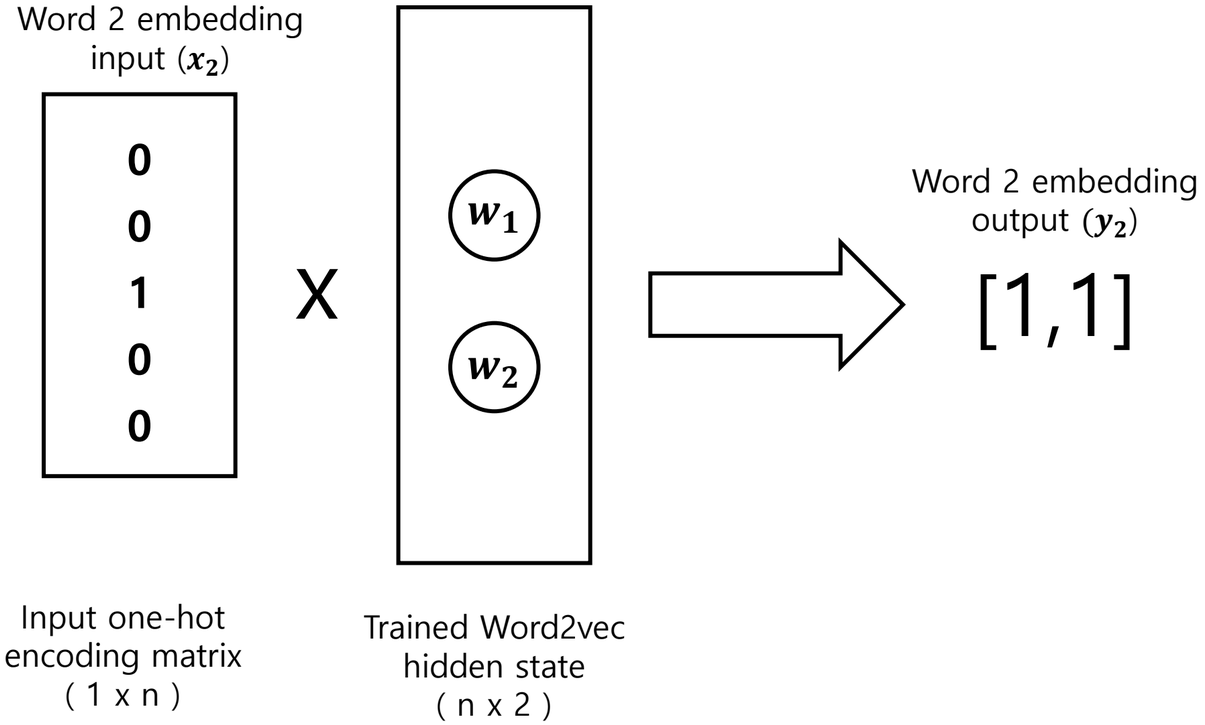
Word	Neighbor
King	Brave
King	Man
Brave	Man
brave	king
Man	brave
Man	brave
Queen	Beautiful
Queen	Woman
Beautiful	Queen
Beautiful	Woman
Woman	Queen
woman	beautiful

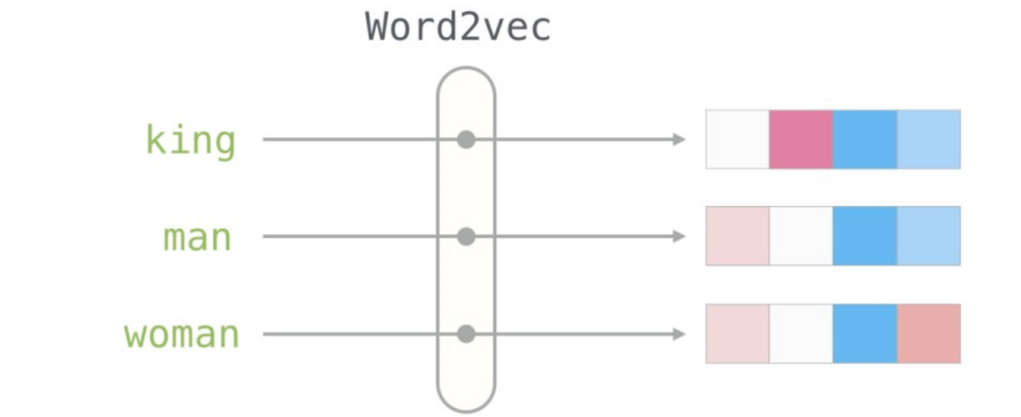
각 word,neighbor을 one-hot encoding 수행 그 결과를 Word2vec의 hidden state를 구하기 위한 input과 output으로 사용

Window size = n : 양 옆으로 n개의 word를 neighbor로 잡는 파라미터 값

02 Related Works

Word2vec





이러한 식으로 embedding을 시킴 -> hidden state는 각 words의 neighbor을 학습 시켜서 생성

n x 2인 이유는 embedding한 값을 2차원 벡터로 표현하기 위함

n : the number of elements of one-hot encoding

02 | Related Works

Glove

- Word2vec의 한계점을 해결하기 위한 해결책
  - Word2vec은 NN 구조로 문장에서 단어의 개수가 많아지면 연산량이 너무 많아짐
  - NN 구조로 이루어져서 반복되는 단어들(a,the 등)의 영향력이 너무 커짐

- 이를 해결하기 위해 동시 등장 행렬(Co-occurrence Matrix)을 사용
  - 동시 등장 행렬이란, 행과 열에는 문장들에서 나온 단어들을 두고, 중심 단어(central word) i를 중심으로 window size내에서 주변 단어(context word) k의 등장 횟수를 symmetric matrix로 표현한 것.

뒷 슬라이드에 동시 등장 행렬 예시 존재

Paper에서 제시한 동시 등장 행렬 확률 표

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

$P(k\ |ice)$: *ice가 들어가 있는 문장 중에서*
k = soild가 들어가 있을 확률이 약 0.00019
k = Gas가 들어가 있을 확률이 약 0.000066
k = Water의 확률이 약 0.003
k = Fashion의 확률이 약 0.000017

$P(k\ |steam)$: *steam이 들어가 있는 문장 중에서*
k = soild가 들어가 있을 확률이 약 0.000022
k = Gas가 들어가 있을 확률이 약 0.00078
k = Water의 확률이 약 0.0022
k = Fashion의 확률이 약 0.000018

$P(k\ |ice)/P(k\ |steam)$:
k = soild 일 때 1보다 큰 8.9
k = Gas 일 때 1보다 작은 0.085
k = Water 일 때 1보다 큰1.36
k = Fashion 일 때 1보다 작은 0.96

위의 확률표에 대한 해석은 마지막 나눈 값이 중요 -> 당연히 soild(단단한)은 steam보다는 ice랑 관련이 많을 테니 8.9가 출력이 되며 $P(k\ |ice)$가 더 크게 나오고 , Gas(가스)는 ice보다는 steam이랑 관련이 더 많을 테니 0.085로 $P(k\ |steam)$ 가 더 크게 나오는 것이다.

02 Related Works

Glove

EX) I like deep learning
I like NLP
I enjoy flying

해당 문장들의 동시 등장 행렬

카운트	I	like	enjoy	deep	learning	NLP	flying
I	0	2	1	0	0	0	0
like	2	0	0	1	0	1	0
enjoy	1	0	0	0	0	0	1
deep	0	1	0	0	1	0	0
learning	0	0	0	1	0	0	0
NLP	0	1	0	0	0	0	0
flying	0	0	1	0	0	0	0

- Glove의 main idea : embedding된 중심 단어와 주변 단어 벡터의 내적이 전체 코퍼스에서의 동시 등장 확률이 되도록 만드는 것

02 | Related Works

Glove

- Glove의 main idea : embedding된 중심 단어와 주변 단어 벡터의 내적이 전체 코퍼스에서의 동시 등장 확률이 되도록 만드는 것

$$dot\ product(w_i\ \tilde{w}_k) \approx\ log\ P(k \mid i) = log\ P_{ik}$$

위의 main idea를 해결하기 위한 Loss function -> 우리가 optimizing 해야 하는 목적 함수

$$Loss\ function = \sum_{m,n=1}^{V} (w_m^T \tilde{w}_n + b_m + \tilde{b_n} - logX_{mn})^2$$

Loss function 및 main idea에서의 변수들의 정의

X : 동시 등장 행렬(Co-occurrence Matrix)

X_{ij} : 중심 단어 i가 등장했을 때 윈도우 내 주변 단어 j가 등장하는 횟수

$X_i : \sum_j X_{ij}$: 동시 등장 행렬에서 i행의 값을 모두 더한 값

$P_{ik} : P(k \mid i) = \frac{X_{ik}}{X_i}$: 중심 단어 i가 등장했을 때 윈도우 내 주변 단어 k가 등장할 확률

Ex) P(solid I ice) = 단어 ice가 등장했을 때 단어 solid가 등장할 확률

$\frac{P_{ik}}{P_{jk}}$: P_{ik}를 P_{jk}로 나눠준 값

Ex) P(solid I ice) / P(solid I steam) = 8.9

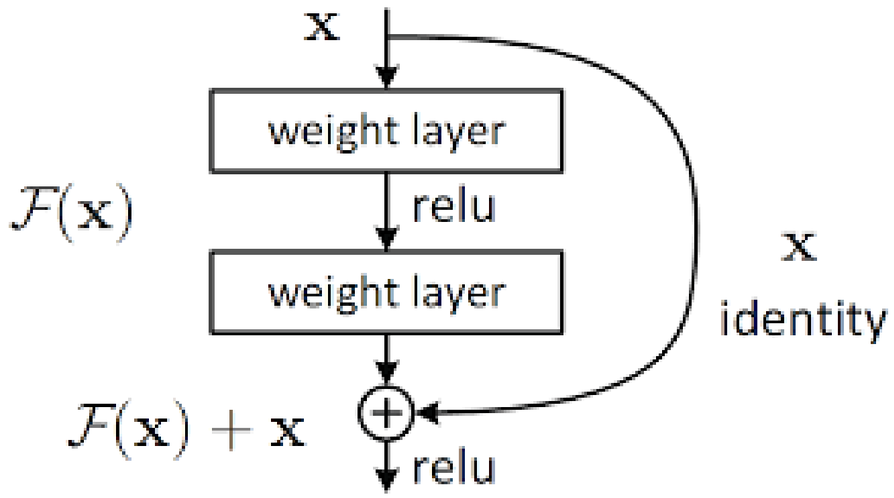
w_i : 중심 단어 i의 임베딩 벡터

$\tilde{w}_k$: 주변 단어 k의 임베딩 벡터

02 Related Works

Residual connection

- Neural Network 학습시 발생되는 gradient vanishing,exploding 문제의 해결책



Resdiual connection은 저번주 CV session에서 진행한 바 있어 간단하게 언급만 하고 넘어감

H(x)=F(x)+x 인 상황에서 residual 의 main idea는 H(x)가 x로 수렴하는 즉, F(x)가 0으로 수렴하도록 만드는 것이 residual connection의 목적

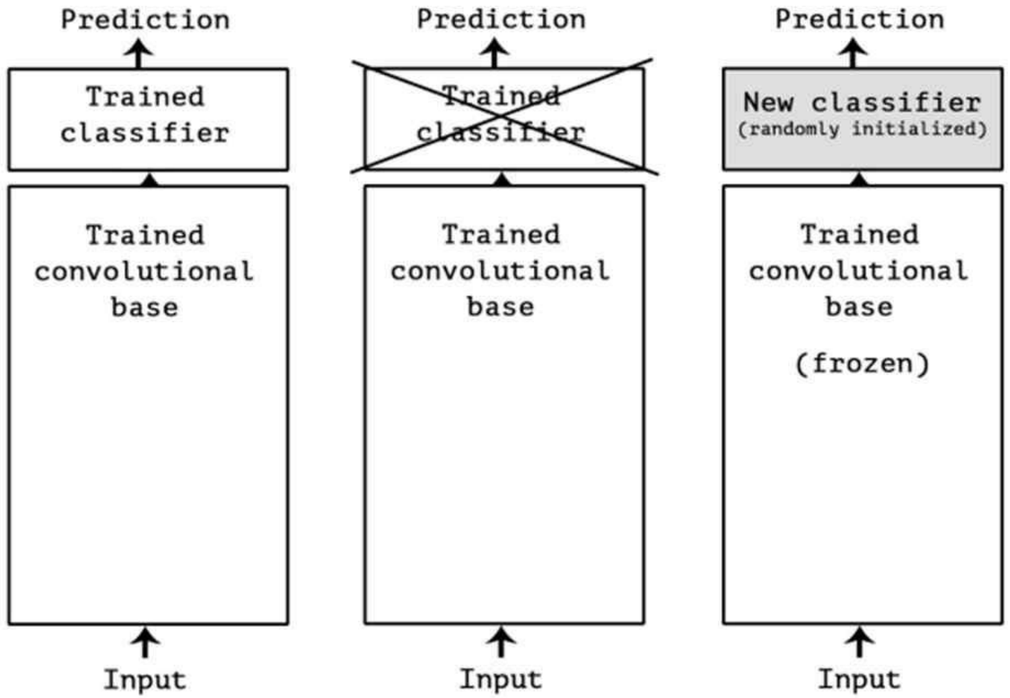
간단히 말하면 딥러닝 학습시 몇 개의 convolutional layer를 지난 후의 output data를 초기의 input data와 connection 하여서 최대한 초기의 input값 x를 보존시키는 것

02 Related Works

Pre-trained model

- Pre-trained model(사전 훈련 모델) (=전이 학습)

- 기존에 알고 있는 다른 지식(source domain)을 통해 새로운 문제(target domain)를 해결하는 학습 방법
- 보다 적은 데이터의 양으로 좋은 모델의 성능을 얻을 수 있고 학습 시간을 줄일 수 있음



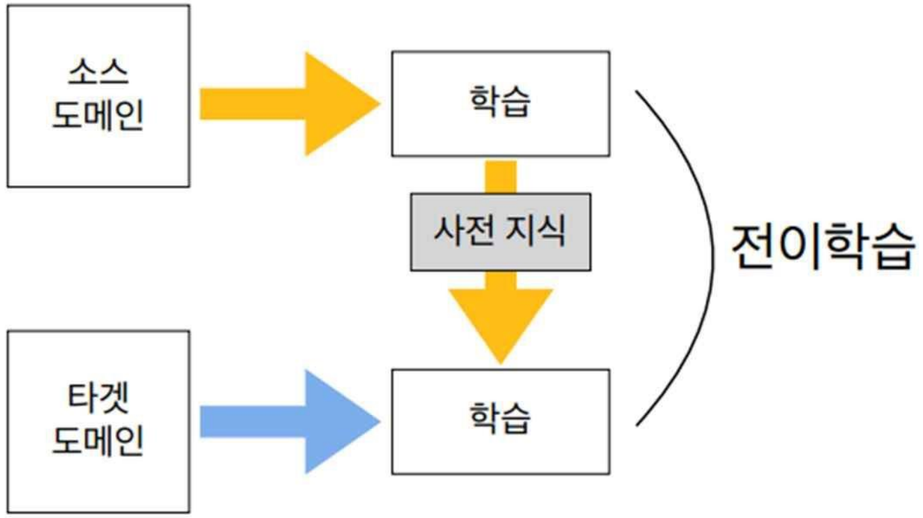
- 왼쪽 그림에서

첫번째는 사전 학습을 진행 중,

두번째는 사전 학습을 마친 상태,

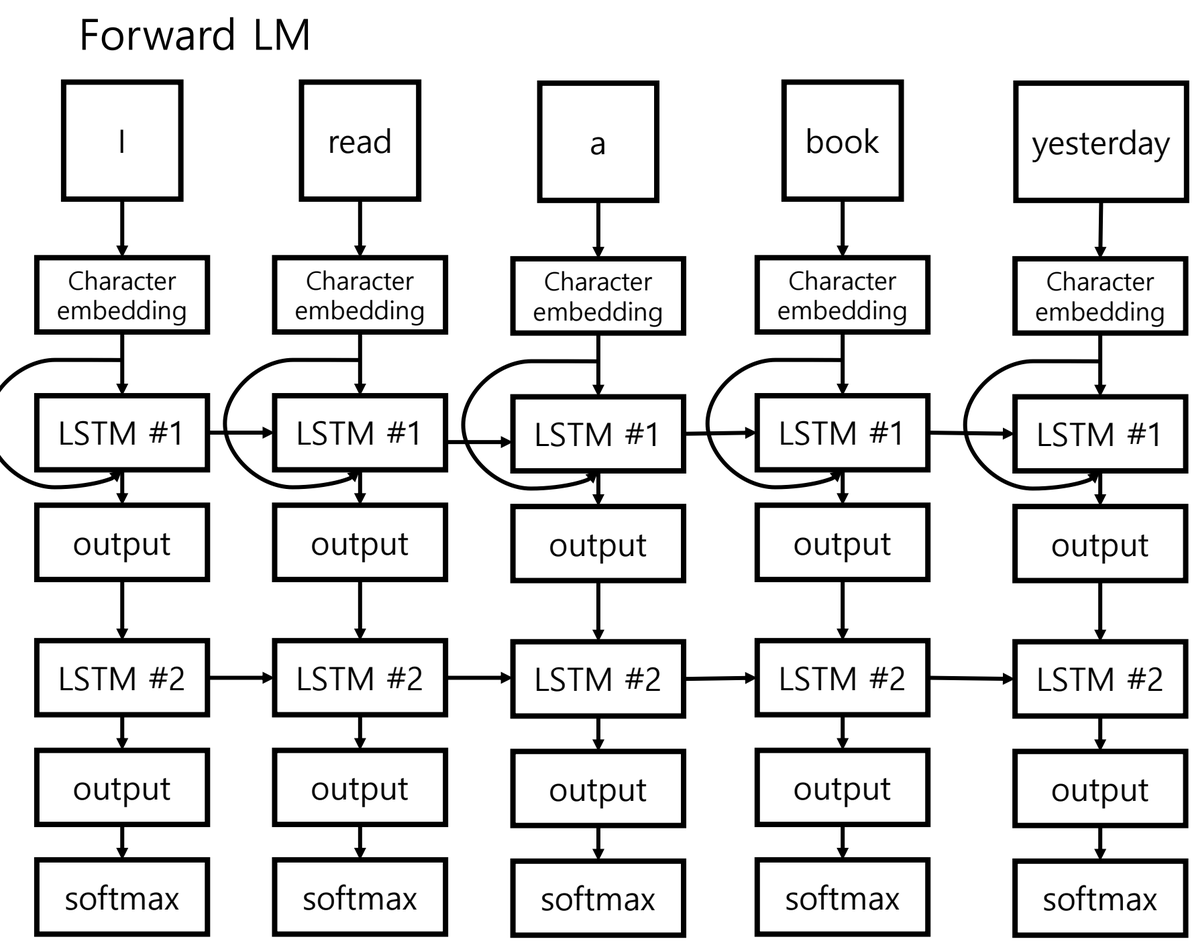
세번째는 사전 훈련 모델에 전이 학습을 진행 시키는 상태 ->
첫번째에서 학습한 모델을 토대로 New classifier를 output으로 사용

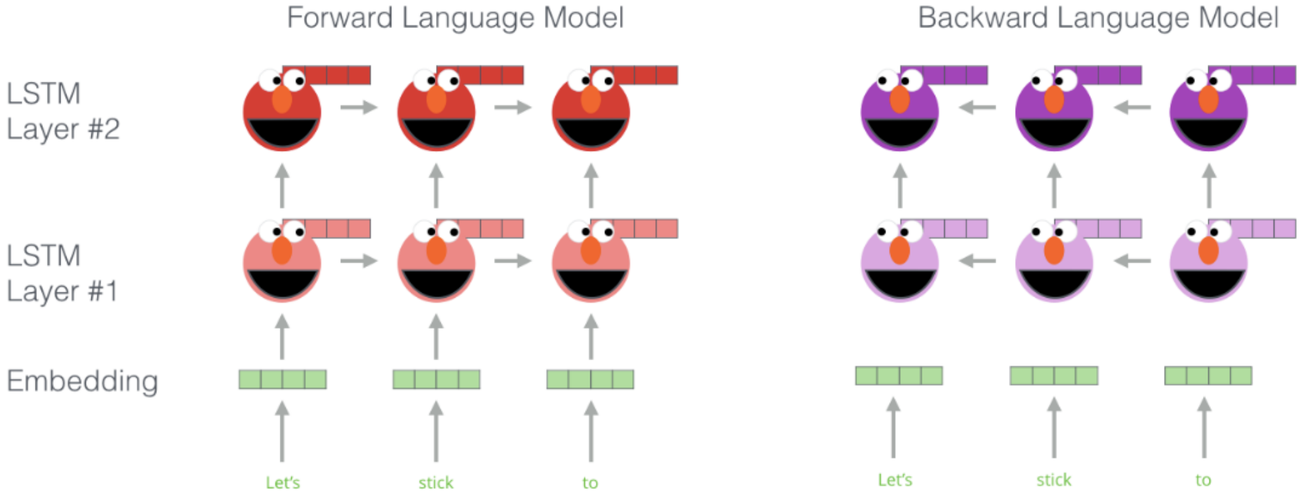
- 사전 훈련 및 전이 학습 과정



03 Proposed Method

Bidirectional LSTM (LM)

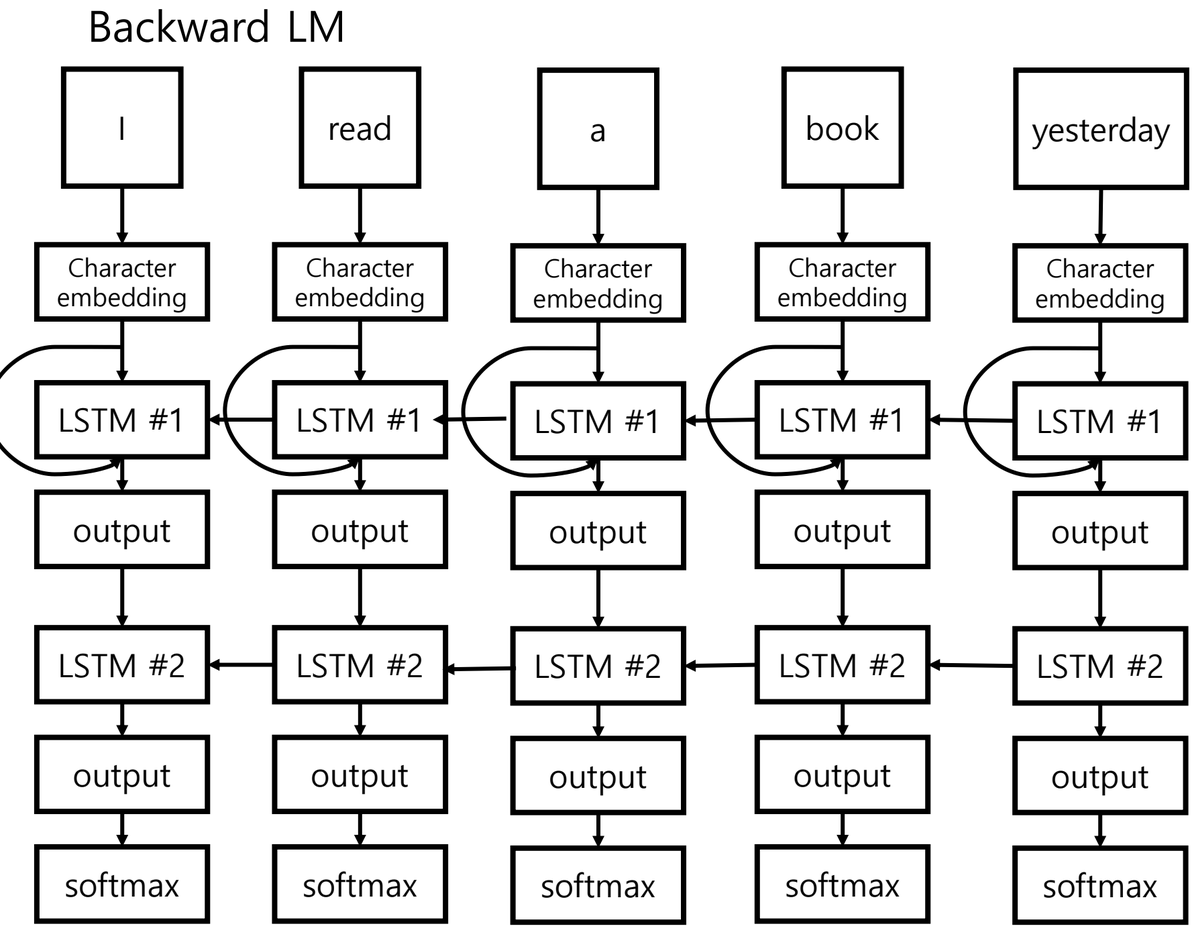




오른쪽은 forward LM으로 순차적으로 학습시 read의 뜻이 yesterday까지 가기 전까지는 의미가 현재의 read인지 과거의 read인지 모름
하지만, 문장의 진짜 의미는 과거의 read
이러한 문제점을 더 쉽게 해결하기 위해 backward LM을 사용해서 bidirectional LSTM(LM)을 고안해냄

03 Proposed Method

Bidirectional LSTM (LM)

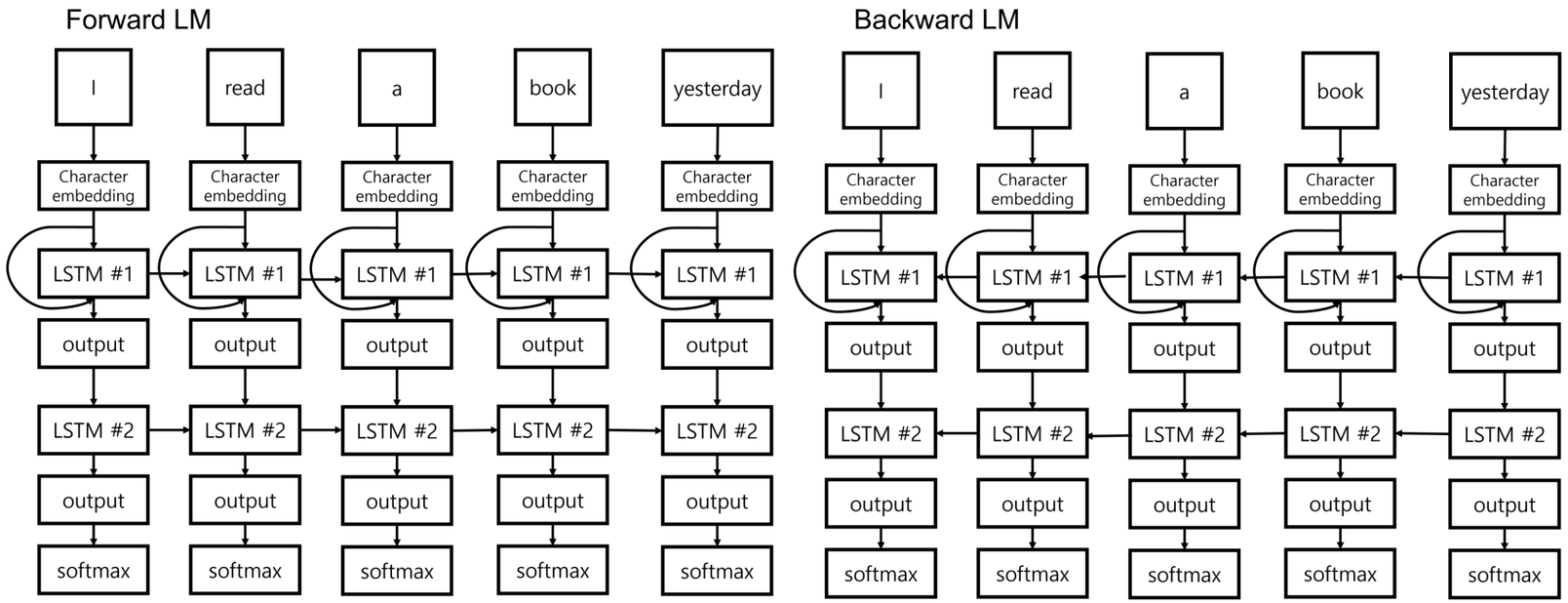


2개의 양방향 LSTM을 사용해서 모델을 학습시킴
-> forward LM은 문장을 순서대로 학습하지만 backward LM은 역방향(reverse)로 문장을 해석해서 해당 문장의 context를 더 정확하고 빠르게 학습할 수 있음

Backward LSTM이 마지막 문장부터 시작해서 그 전 단어를 학습하면서 해당 문장의 시기가 yesterday로 인해 과거형임을 학습함 -> 문장 앞쪽에 있는 read가 과거의 read임을 알게됨

03 Proposed Method

Bidirectional LSTM (LM)



각 forward LSTM, backward LSTM 에서 LSTM layer를 2개를 사용하고 input data를 일반 embedding이 아닌 Character embedding을 하는 이유는 각 단어들의 문맥 및 의미를 더 자세히 파악하기 위해 사용

04 | Experiment

05 | Conclusion

Q & A

감사합니다