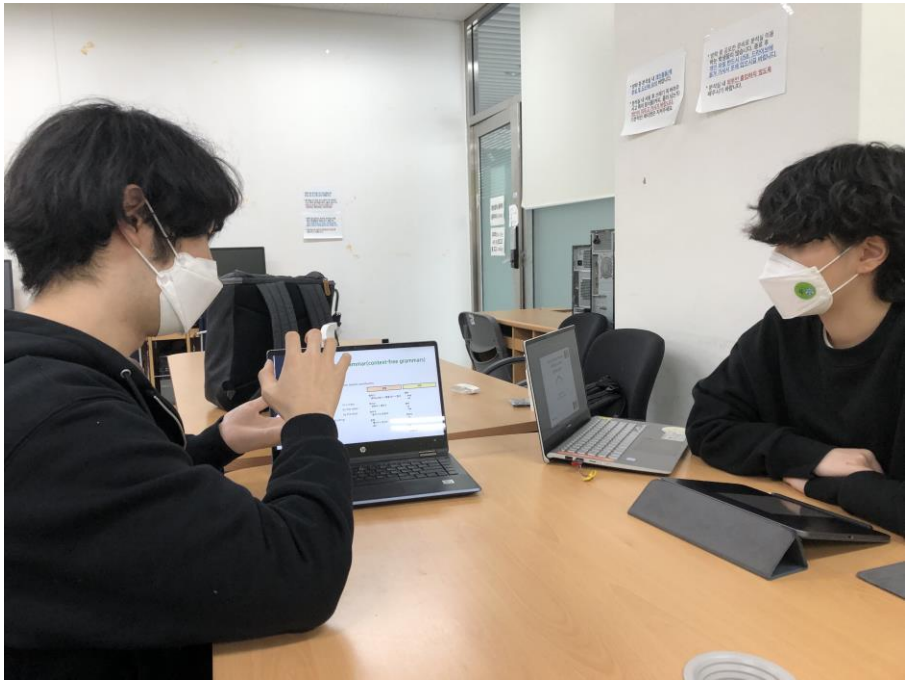
CUAI 스터디 CS224n 1팀

2022.05.03

발표자 : 김서린

스터디원 소개 및 만남 인증



스터디원 1 : 김병찬(통계학과)

스터디원 2 : 김서린(응용통계학과)

스터디원 3 : 이재용(통계학과)

22.03.31. 15:00 – 17:00
응용통계학과 분석실에서 진행

스터디 스케줄

Date	Week	Topic
Mar 17, 2022	1	Lecture 1 – Introduction and Word Vectors
Mar 17, 2022	1	Lecture 2 - Neural Classifiers
Mar 17, 2022	1	Lecture 3 - Backprop and Neural Networks
Mar 31, 2022	2	Lecture 4 - Dependency Parsing
Mar 31, 2022	2	Lecture 5 - Language Models and RNNs
May 12, 2022	4	Lecture 6 - Simple and LSTM RNNs
May 12, 2022	5	Lecture 7 - Translation, Seq2Seq, Attention
미정	6	Lecture 9 - Self- Attention and Transformers

Lecture 4 :
Dependency Parsing

Syntactic Structure : Consistency and Dependency

1) Consistency Parsing (Phrase structure grammar, CFGs)

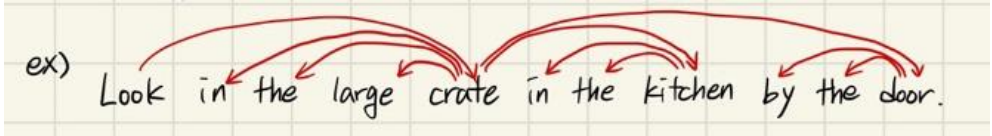
- word → phrase → bigger phrase
- 동사구, 명사구, 전치사구 등으로 묶어서 문장의 구조 파악

ex) Grammar

	Lexicon
NP → Det (Adj)* N (PP)	N → dog, cat
PP → P NP	Det → a, the
VP → V PP	P → in, on, by
S → NP VP	V → talk, walked

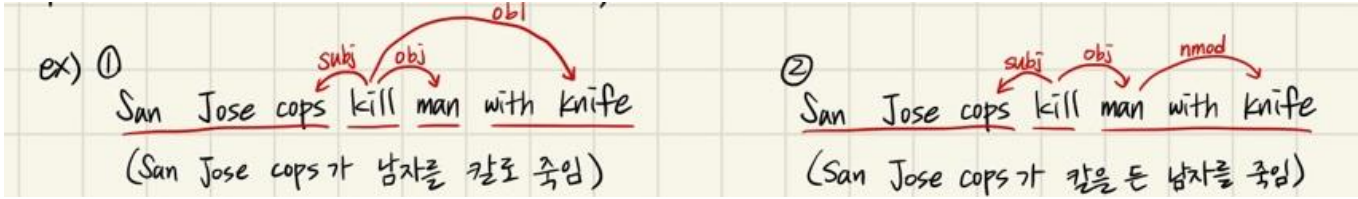
2) Dependency Parsing

- 문장에서 각 단어에 대해 어떤 단어가 그것을 수식하는지 설명하는 것이 목표



Prepositional Phrase Attachment Ambiguity

San Jose cops kill man with knife

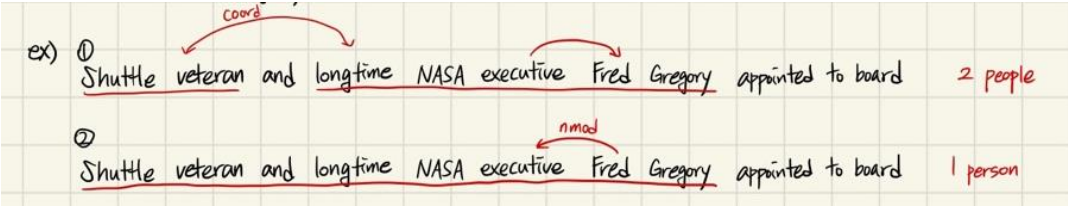


1) PP attachment ambiguities multiply

- 문장 끝에 k개 전치사구 있으면 문장은 Catalan numbers만큼 parse할 수 있음
- Catalan number = (2n)! / (n+1)!n!

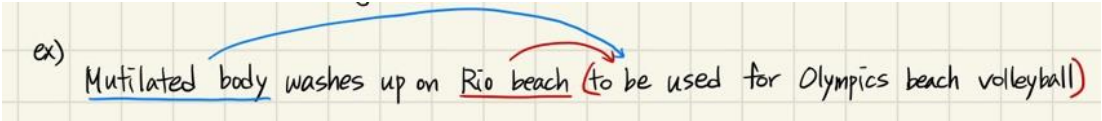
Other Ambiguities

1) Coordination scope ambiguity



2) Adjectival/Adverbial modifier ambiguity

3) Verb Phrase attachment ambiguity



Dependency Grammar and Dependency Structure

1) ROOT에 의존하는 단어는 단 하나
2) A→B, B→A 등의 사이클이 없어야 함
3) Non-projective sentence (화살표가 겹치면 안됨)

Basic Transition-based Dependency Parser

1) Initialization
- Stack (sigma) : ‘ROOT’
- Buffer (beta) : parse하려는 문장의 단어들
- A : dependency arcs

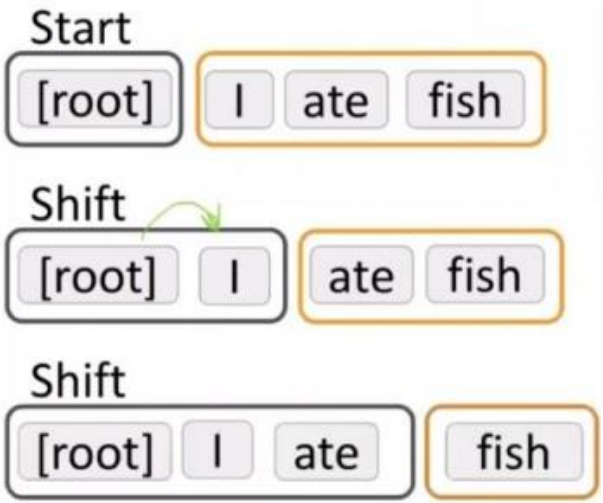
2) Shift
- 다음 단어를 스택에 넣기

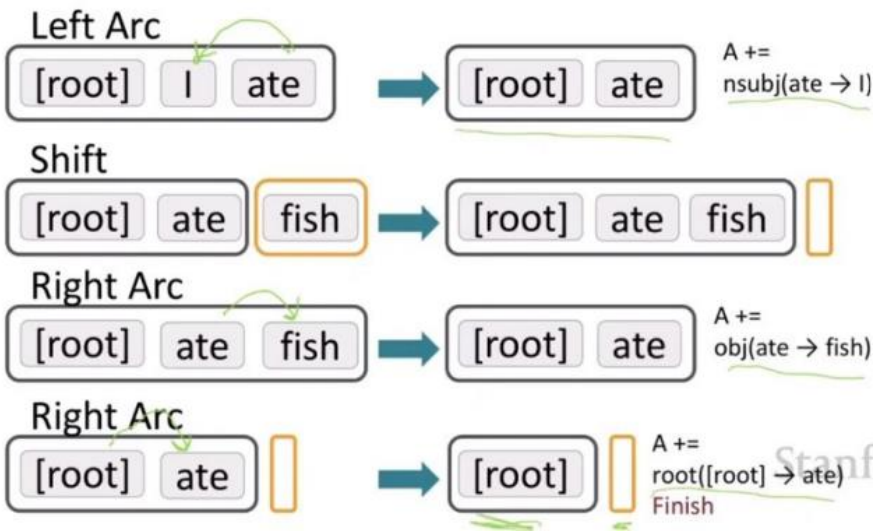
3) Left-Arc or Right-Arc
- 스택에서 단어 두 개 가져와서 의존관계(수식관계) 파악 (Left or Right)
- A에 두 단어의 문법적 관계 + Left or Right Arc 추가
- 수식받은 단어는 스택에서 사라짐

4) Final
- Stack (sigma) : ‘ROOT’
- Buffer (beta) : 공집합

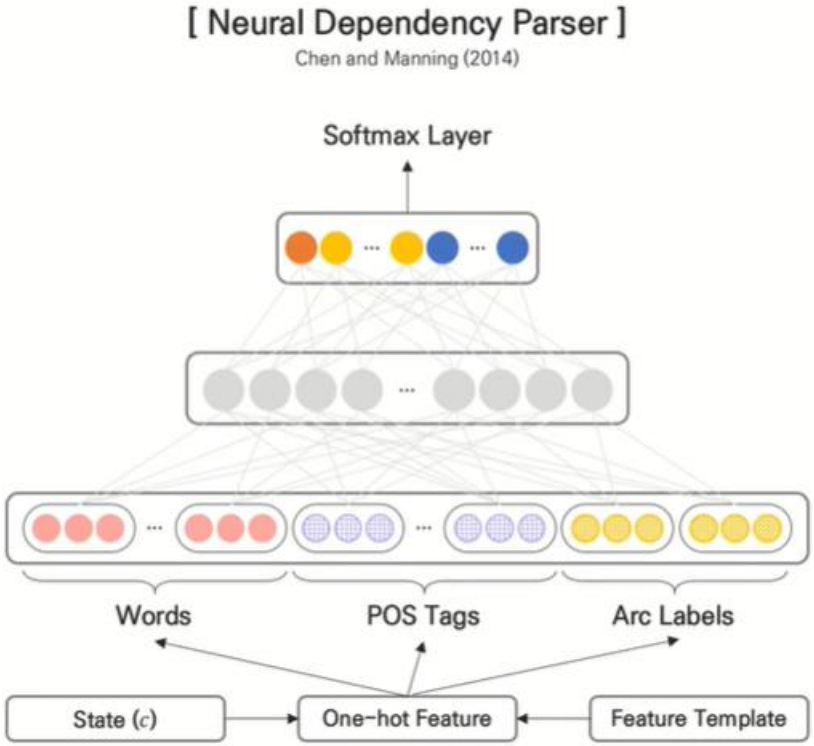
Arc-standard Transition-based Parser

I ate fish





Neural Dependency Parser



4) Softmax Layer

3) Hidden Layer

2) Feature Embedding

1) Feature Selection

Neural Dependency Parser

1) Feature Selection

- STACK, BUFFER의 top 3 단어 (6개)
  : s1, s2, s3, b1, b2, b3

- STACK top 1,2 단어의 1st and 2nd left and right child 단어 (8개)
  : lc1(s1), rc1(s1), lc2(s1), rc2(s1), lc1(s2), rc1(s2), lc2(s2), rc2(s2)

- STACK top 1,2 단어의 (left of left) and (right of right) child 단어 (4개)
  : lc1(lc1(s1)), rc1(rc1(s1)), lc1(lc1(s2)), rc1(rc1(s2))

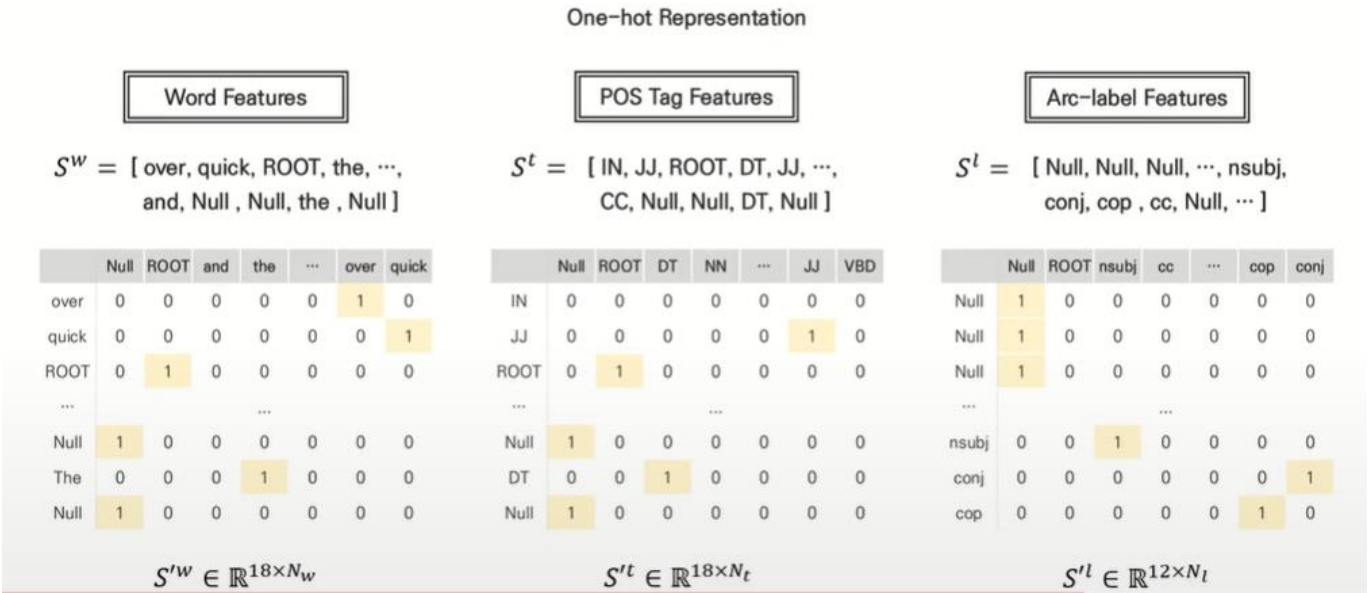
- 선택된 word feature에 해당하는 POS Tag (18개)

- STACK과 BUFFER의 6개 단어를 제외하고 선택된 word에 달린 arc-label (12개)

→ word feature 18개 + POS Tag feature 18개 + Arc-label feature 12개 = 48개 feature

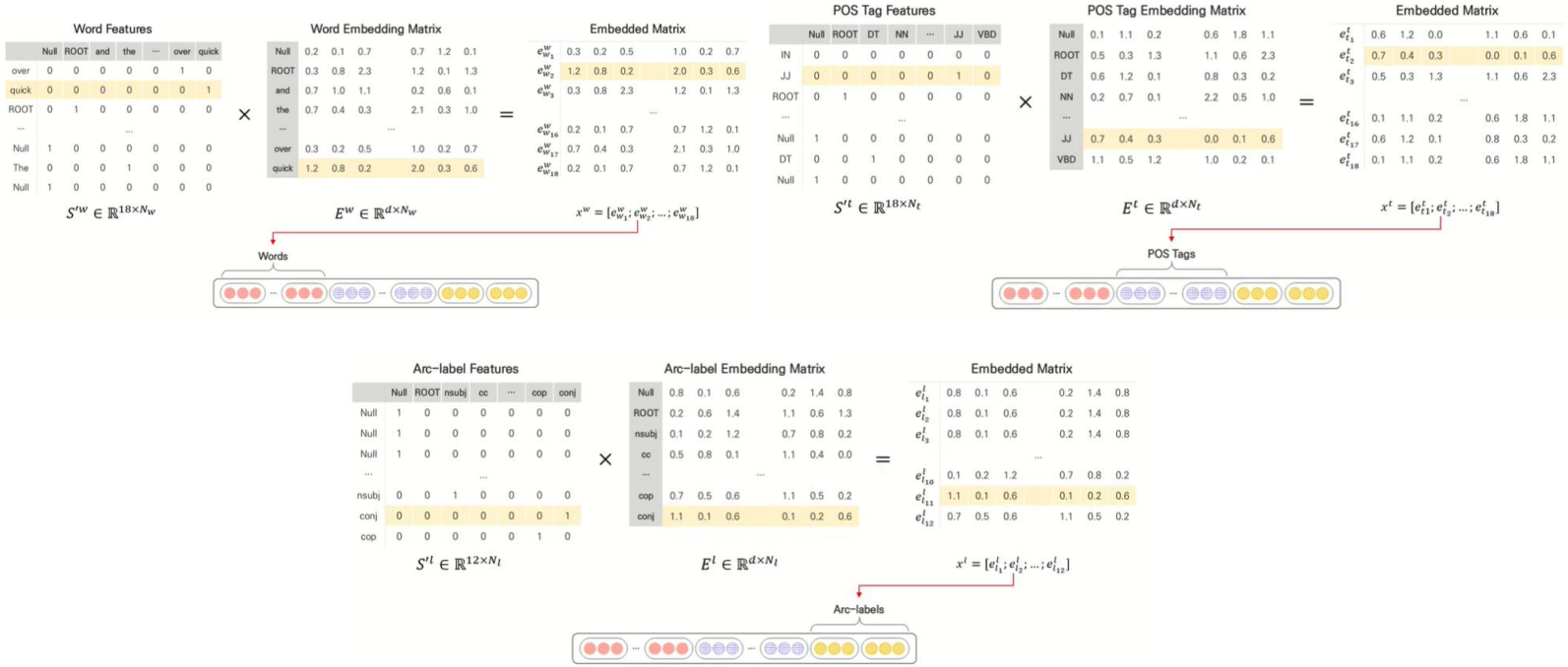
Neural Dependency Parser

1) Feature Selection



Neural Dependency Parser

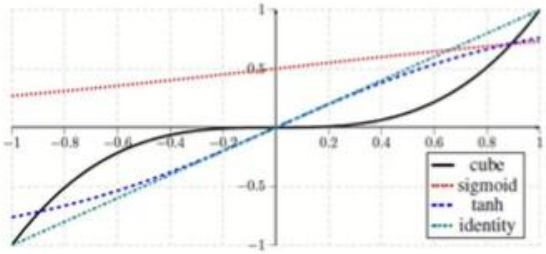
2) Feature Embedding



Neural Dependency Parser

3) Hidden Layer

- 일반적인 feed forward network
- But, ReLU, Sigmoid, Tanh와 같은 일반적인 activation function 사용 X
- 대신 word, POS Tag, Arc-label 간 상호작용 반영할 수 있는 cube function 사용
- 실험 결과 타 non-linearity 대비 우수한 성능



$$h = (W_1[x^w; x^t; x^l] + b)^3$$

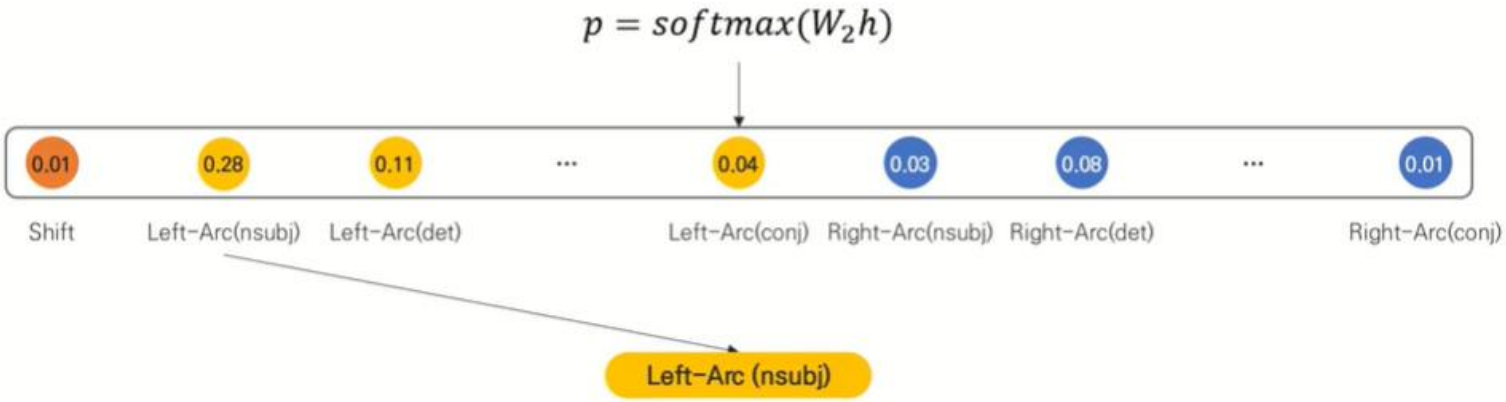
$$= (w_1x_1 + w_1x_2 + \cdots + w_{48}x_{48} + b)^3$$

$$= \sum_{i,j,k} (w_i w_j w_k) x_i x_j x_k + \sum_{i,j} b(w_i w_j) x_i x_j + \cdots$$

각 word, POS tag, arc-label의 조합

Neural Dependency Parser

4) Softmax Layer



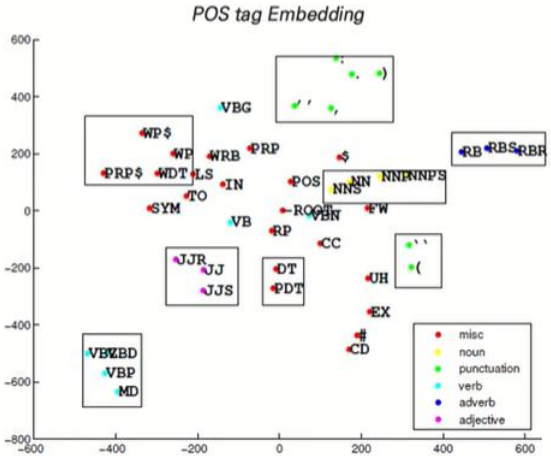
Neural Dependency Parser

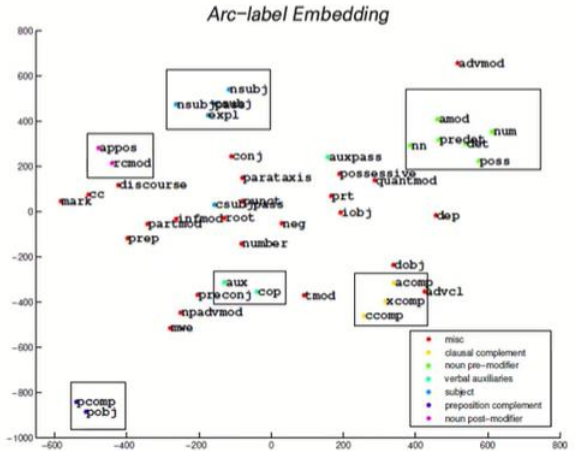
Evaluation Measures

1) UAS (Unlabeled Attachment Score) : Arc 방향만 예측
2) LAS (Labeled Attachment Score) : Arc 방향과 함께 label도 예측

POS and Label Embedding

유사한 요소들끼리 가까이 위치함





Lecture 5 :
Language Models and Recurrent Neural Networks

Language Model

Language Model
- 단어의 시퀀스에 대해, 얼마나 자연스러운 문장인지 확률을 이용해 예측하는 모델

Language Modeling
- 주어진 단어의 시퀀스에 대해 다음에 나타날 단어가 무엇인지 예측하는 작업
- 기계번역, 음성인식, 자동완성 등에 이용

$$P(w_t|w_{t-1}, \dots, w_1)$$

where w_t can be any word in the vocabulary $V = \{w_1, \dots, w_{|V|}\}$

$$P(w_1, \dots, w_T) = P(w_1) \times P(w_2|w_1) \times \cdots \times P(w_T|w_{T-1}, \dots, w_1)$$

$$= \prod_{t=1}^{T} P(w_t|w_{t-1}, \dots, w_1)$$

N-gram Language Models

N-gram Language Models

- Neural Network 이전에 사용되었던 Language model
- 예측에 사용할 앞 단어들의 개수를 정하여 (n개) 모델링

$$P(w_t|w_{t-1},\dots,w_1) \approx P(w_t|w_{t-1},\dots,w_{t-n+1}) \quad \text{(assumption)}$$

prob of a n-gram

prob of a (n-1)-gram

$$= \frac{P(w_t, w_{t-1}\dots,w_{t-n+1})}{P(w_{t-1},\dots,w_{t-n+1})} \quad \text{(definition of conditional prob)}$$

$$\approx \frac{\text{count}(w_t, w_{t-1}\dots,w_{t-n+1})}{\text{count}(w_{t-1},\dots,w_{t-n+1})} \quad \text{(statistical approximation)}$$

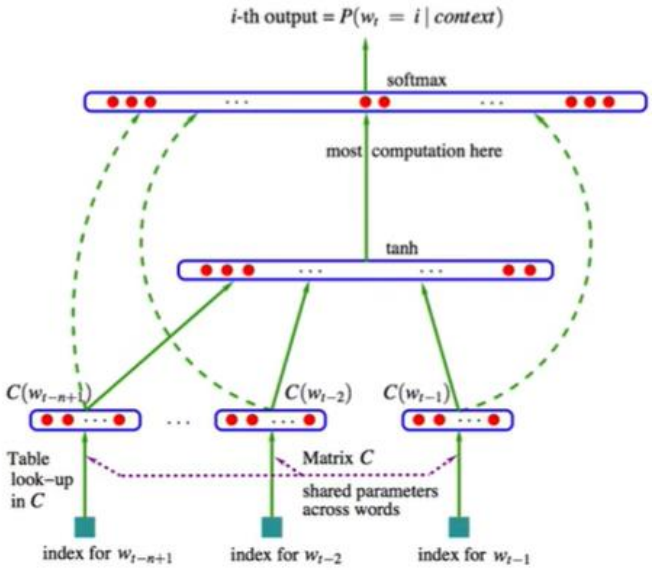
문제점

- Sparsity Problems : n이 커질수록 안 좋아짐. 일반적으로 n<5로 설정. smoothing, backoff
- Storage Problems : n이 커지거나 corpus가 증가하면 모델의 크기가 증가함

Window-based Neural Network Language Model (NNLM)

NNLM

- N-gram 모델의 sparsity problem을 해결하기 위해 제안된 신경망 기반 모델
- Language model이면서 동시에 단어의 ‘distributed representation’을 학습



Window-based Neural Network Language Model (NNLM)

output distribution

$\hat{y} = \text{softmax}(Uh + b_2) \in \mathbb{R}^{|V|}$

hidden layer

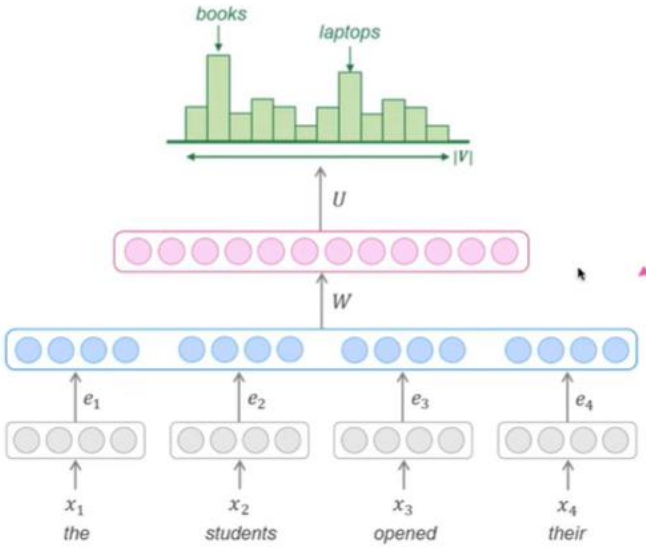
$h = f(We + b_1)$

concatenated word embeddings

$e = [e_1 ; e_2 ; e_3 ; e_4]$

words → one-hot vectors

x_1, x_2, x_3, x_4



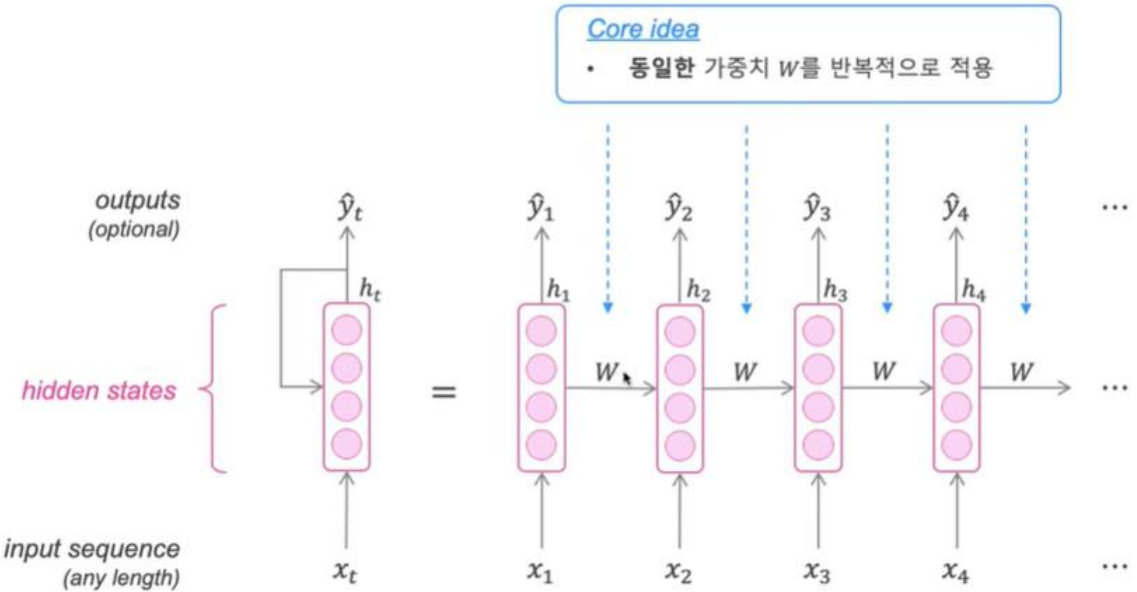
Problems

- Fixed window is too small
- Window가 커질 수록 W도 커짐
  → window 크기의 한계
- x_1과 x_2는 완전히 다른 가중치 W가 곱 해지기 때문에 No symmetry함

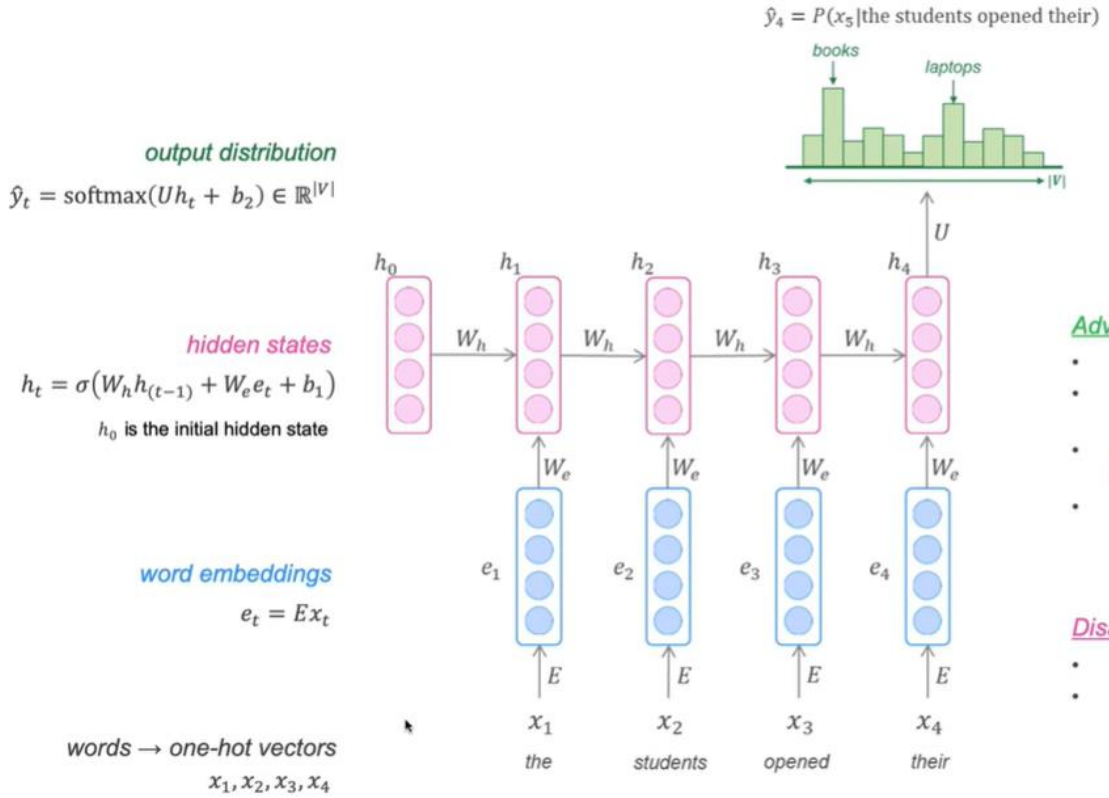
Improvement

- 단어의 embedding을 통한 sparsity problem 해결
- 관측된 *n*-gram을 저장할 필요가 없음

Recurrent Neural Network (RNN)



Recurrent Neural Network (RNN)



Advantages

- 입력(input)의 길이에 제한이 없음
- (이론적으로) 길이가 긴 timestep t 에 대해 처리 가능함
- 입력에 따른 모델의 크기가 증가하지 않음
- 매 timestep t 에 동일한 가중치를 적용하므로 symmetry 함

Disadvantages

- Recurrent 계산이 느림
- 실제로는 길이가 긴 timestep t 에 대해 처리가 안됨

References

https://youtube.com/playlist?list=PLoROMvodv4rOSH4v6133s9LFPRHjEmbmJ
https://youtube.com/playlist?list=PLetSlH8YjIfVdobI2IkAQnNTb1Bt5Ji9U

감사합니다☺