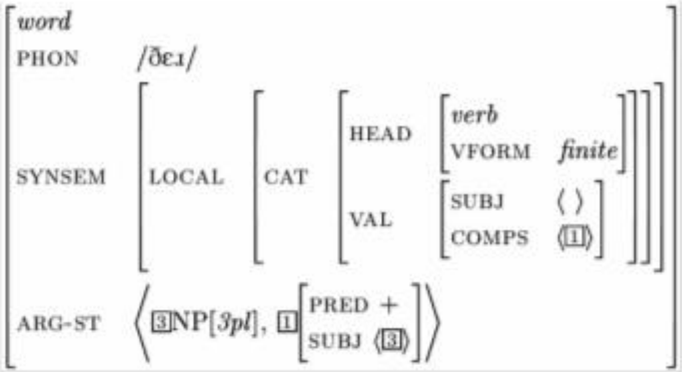
Text Seminar

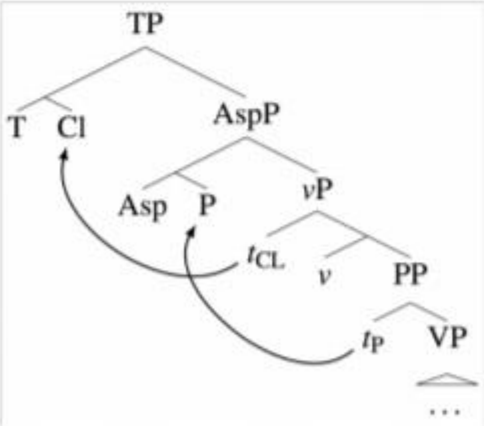
ToBig's 13기 정민준

Constituency Parsing

TreeRNNs

Lecture 18 | Consituency Parsing & Tree RNNs

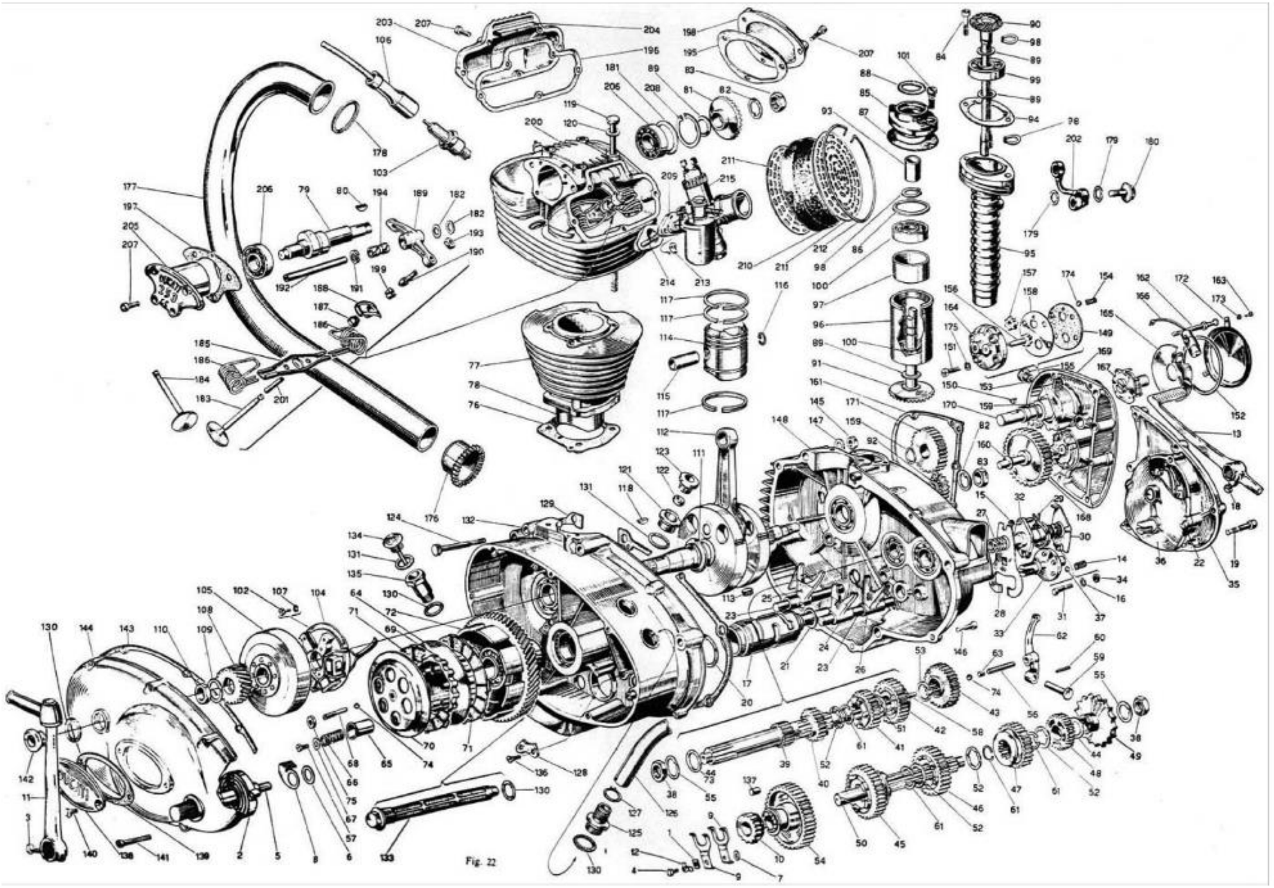




어떻게 하면 큰 구절의 의미를 알아낼 수 있을까?

Lecture 18 | Consituency Parsing & Tree RNNs

Compositionality



Lecture 18 | Consituency Parsing & Tree RNNs

The $\boxed{\text{snowboarder}}$ is leaping over a mogul

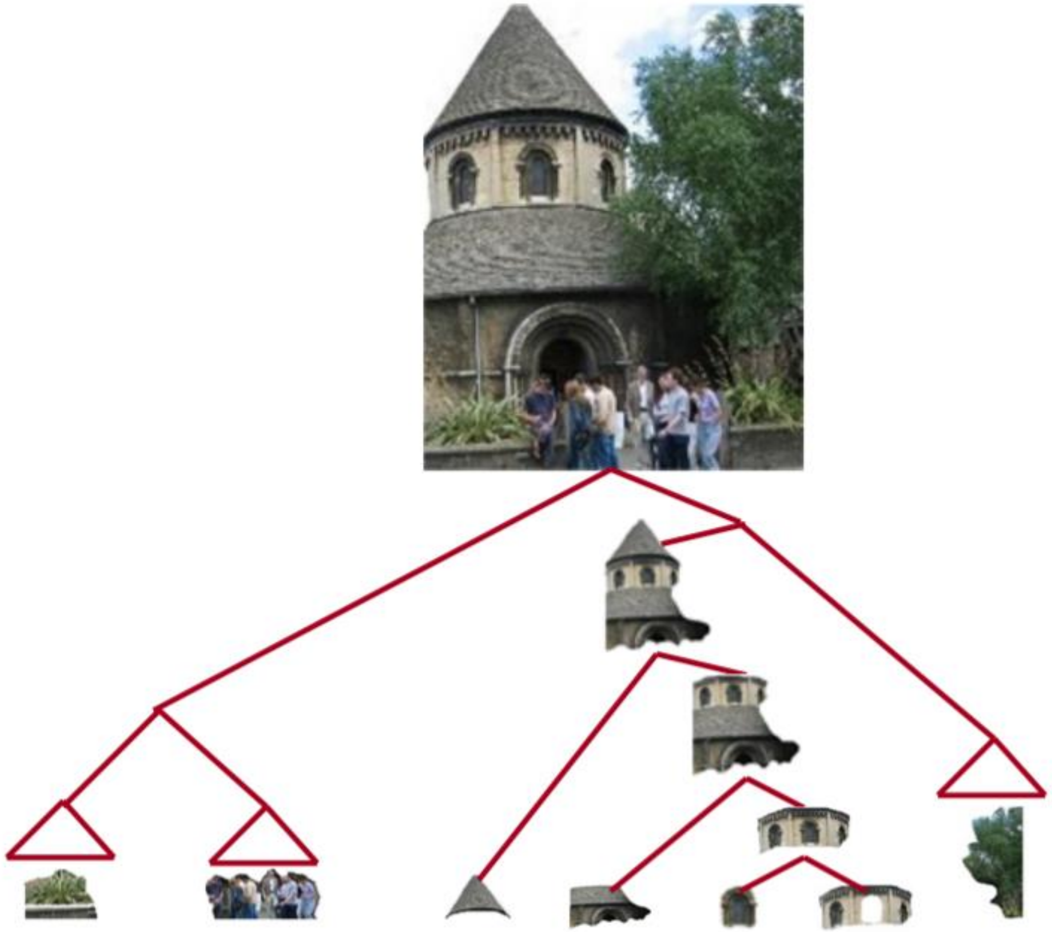
=

$\boxed{\text{A person on a snowboard}}$ jumps into the air

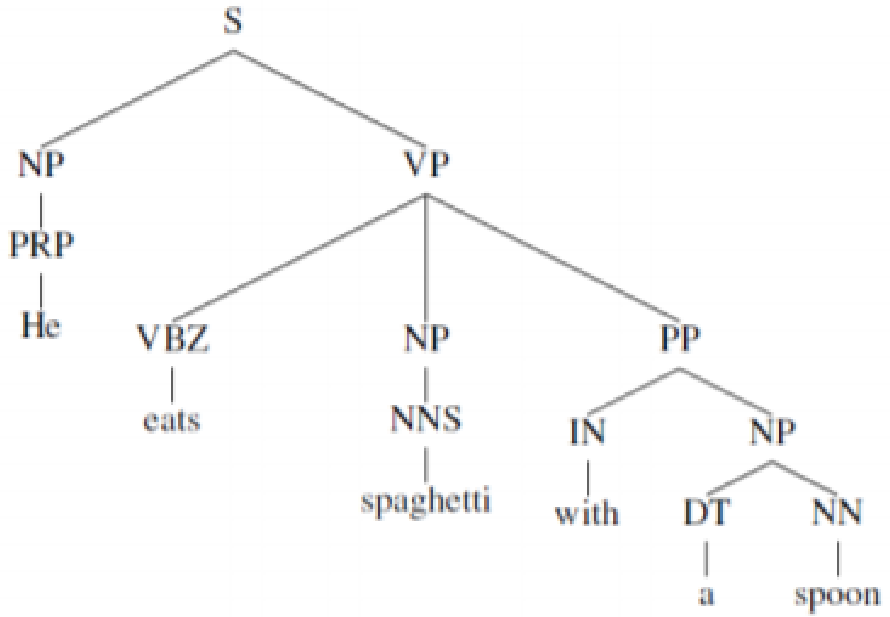
사람들은 큰 텍스트 단위의 의미를 작은 요소의 조합을 통해 의미를 표현한다!

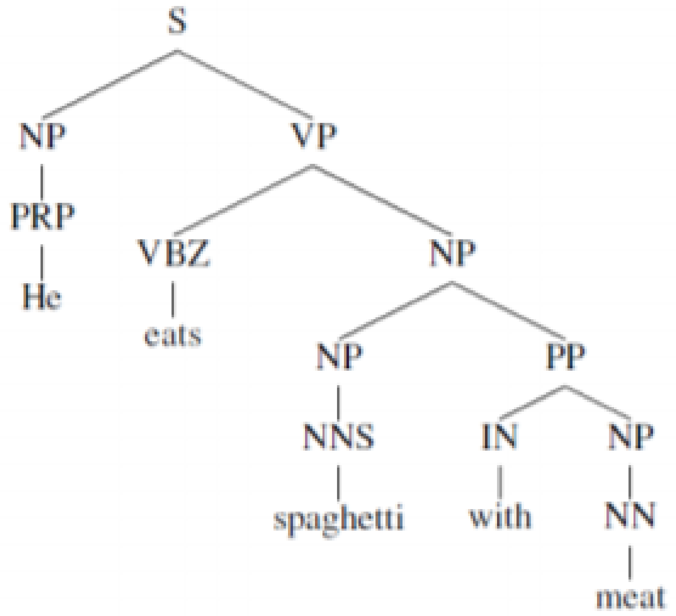
Lecture 18 | Consituency Parsing & Tree RNNs

작은 부분들이 합쳐짐!

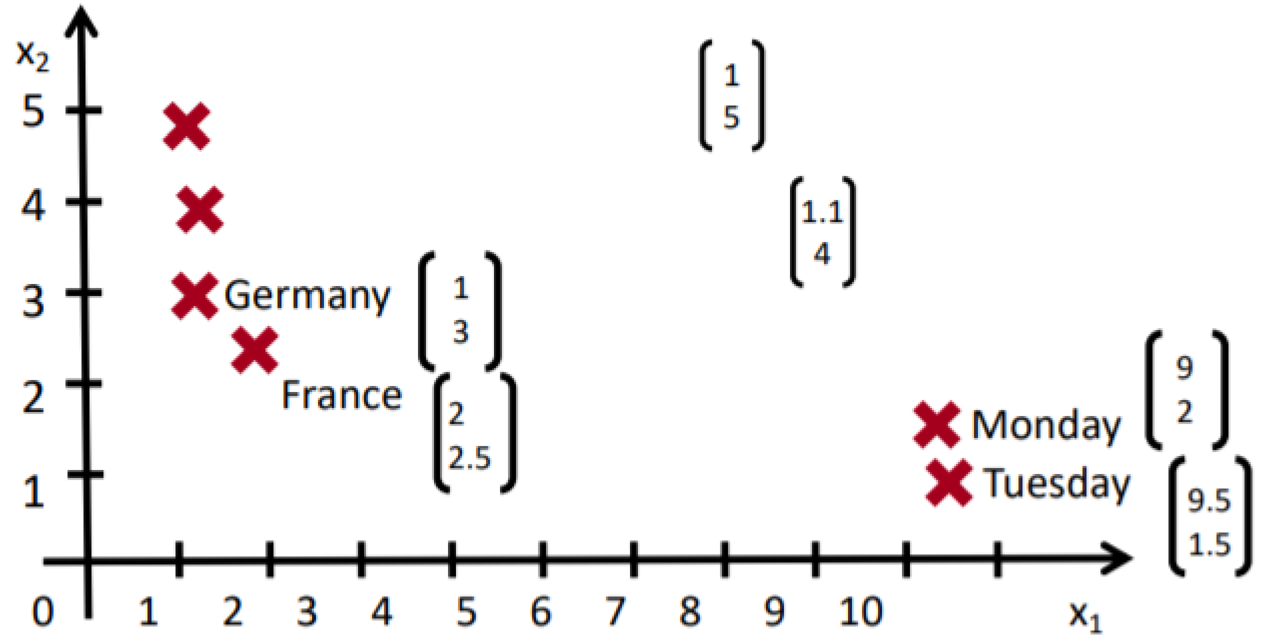


Lecture 18 | Consituency Parsing & Tree RNNs



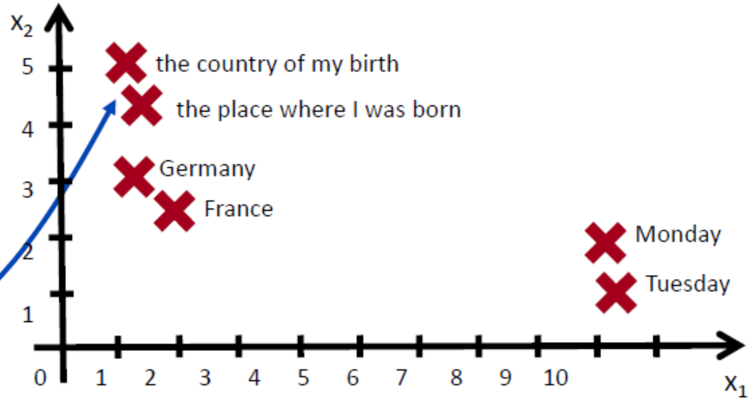


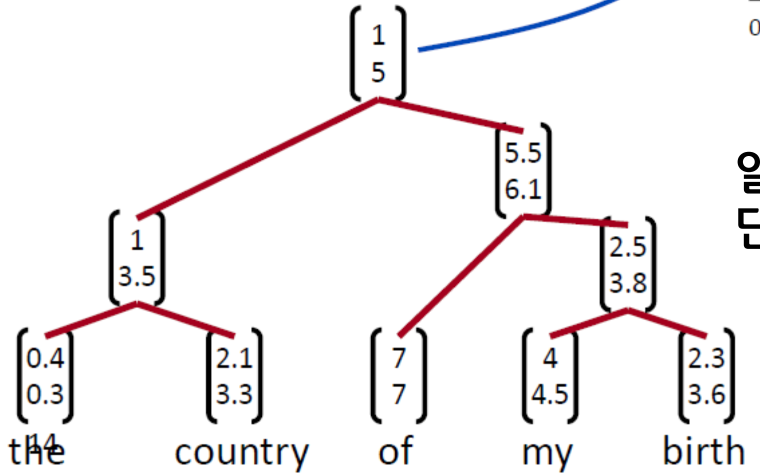
Lecture 18 | Consituency Parsing & Tree RNNs



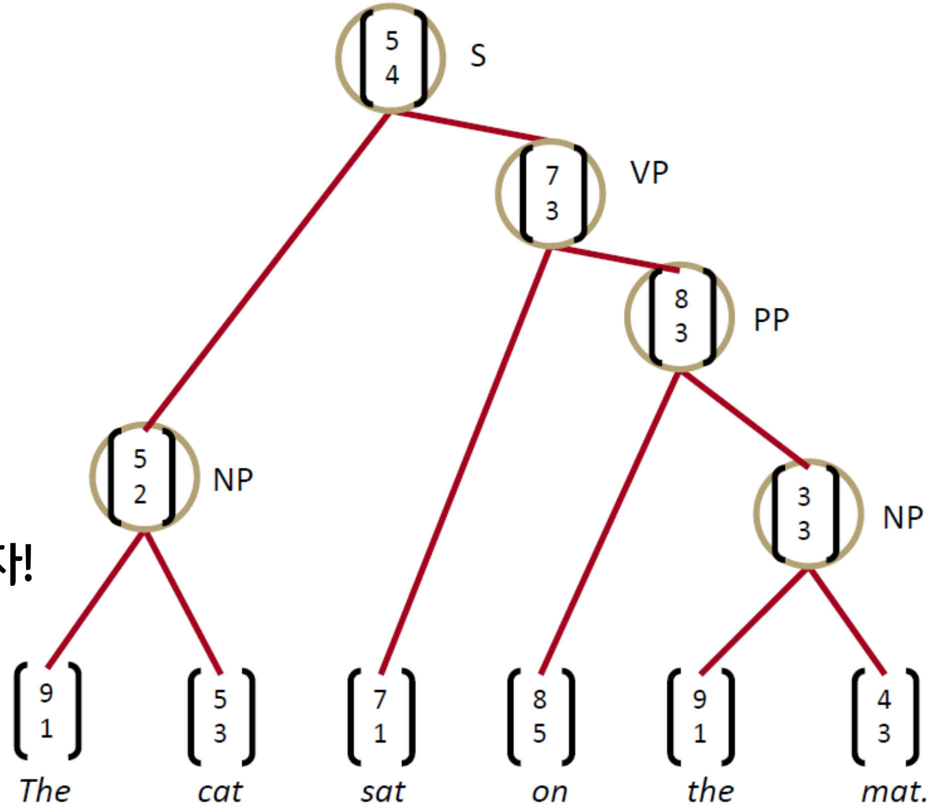
더 큰 구절은 어떻게 매핑시킬까?
ex) the country of my birth, the place where I was born

Lecture 18 | Consituency Parsing & Tree RNNs

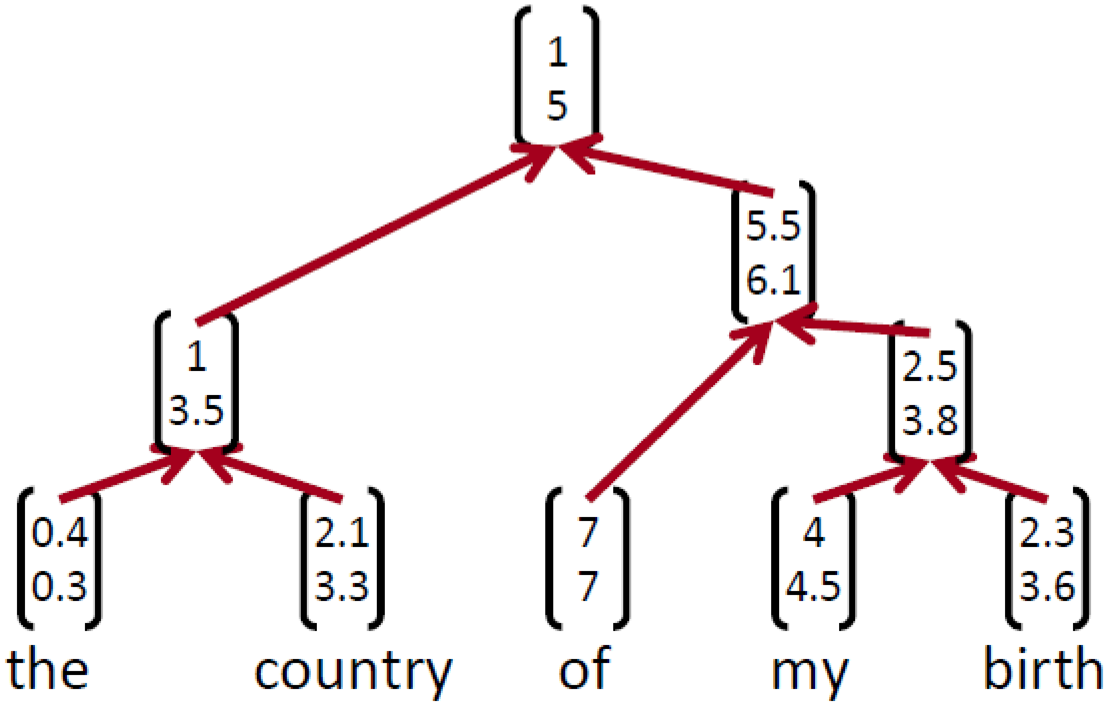


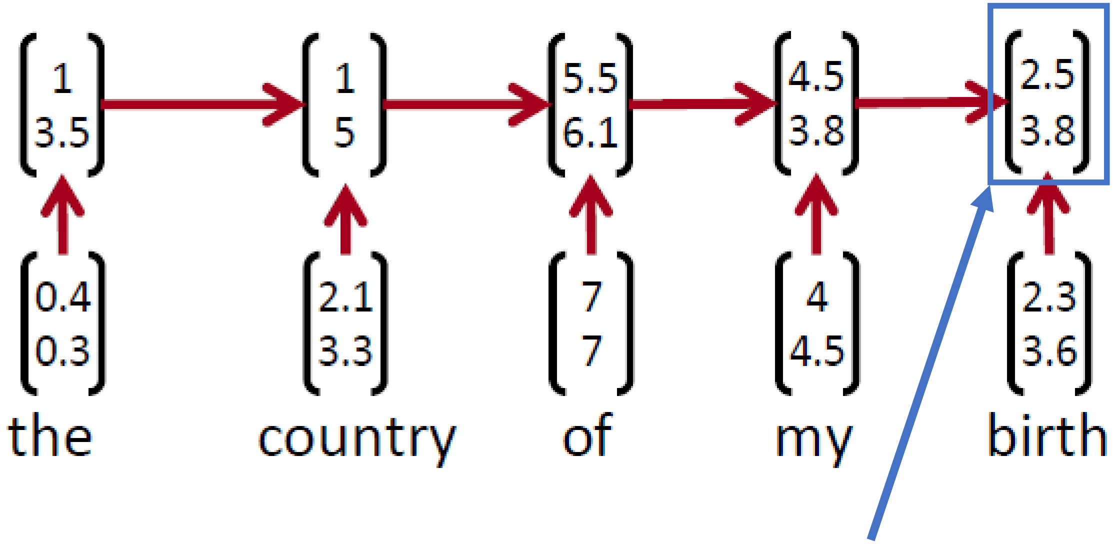


일정한 규칙을 정하여
단어간 조합을 하고 의미 벡터를 만들자!



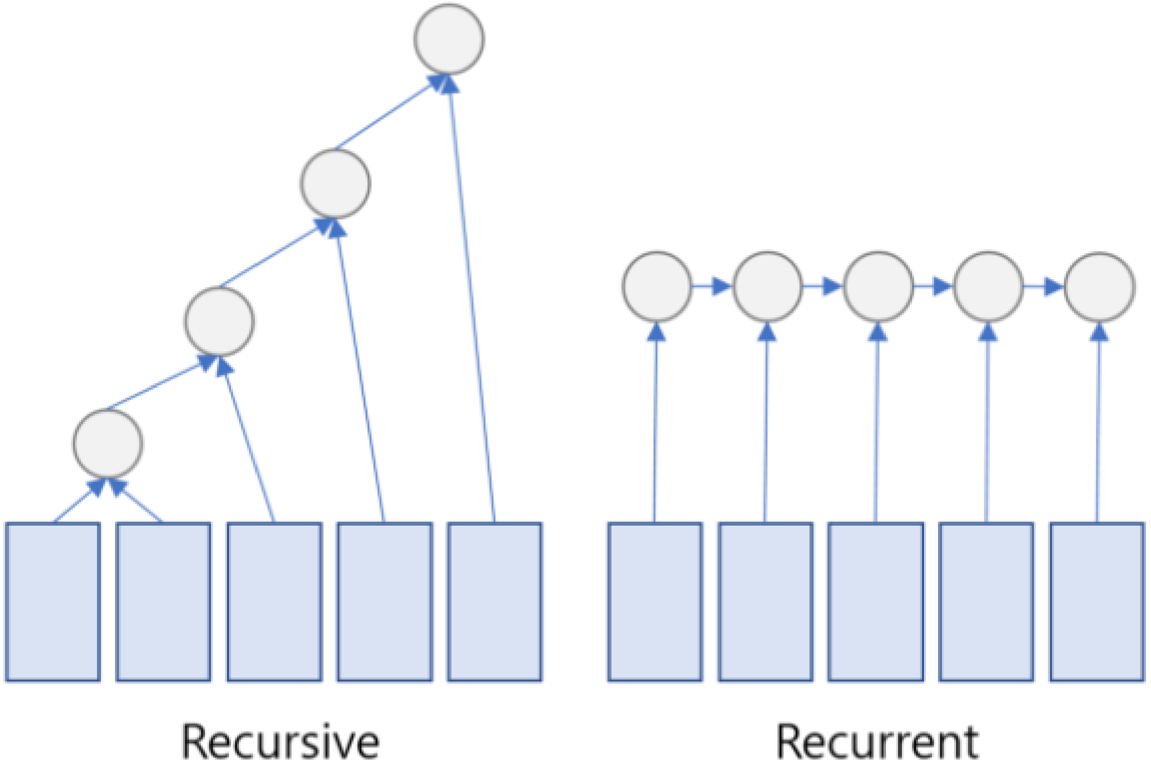
Recursive vs RNN



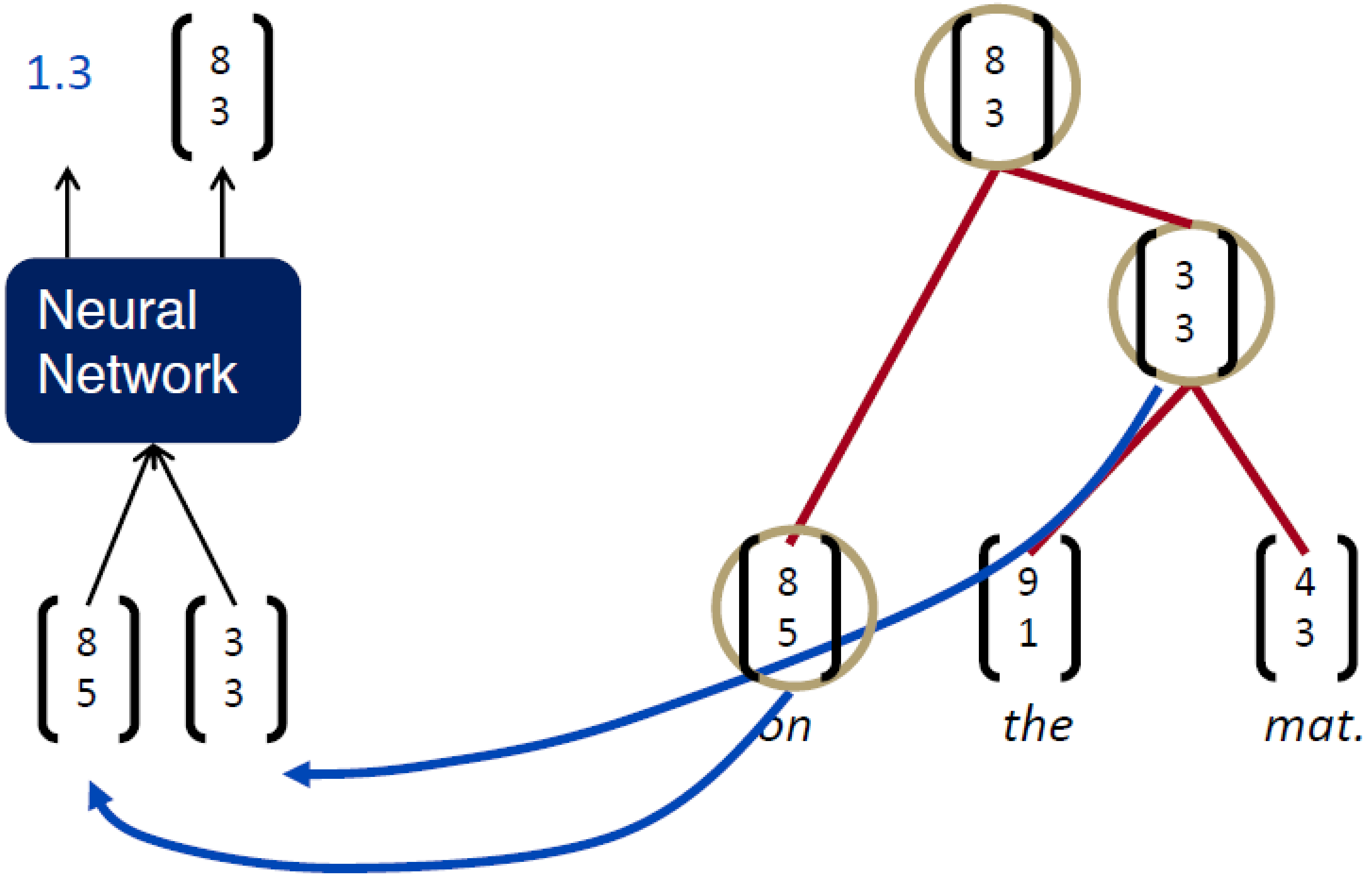


인접 단어를 합친 단어의 의미는 충분히 반영하지 못한다!
주로 마지막 단어 벡터를 주목하는 경향이 있음.

Lecture 18 | Consituency Parsing & Tree RNNs



Lecture 18 | Consituency Parsing & Tree RNNs



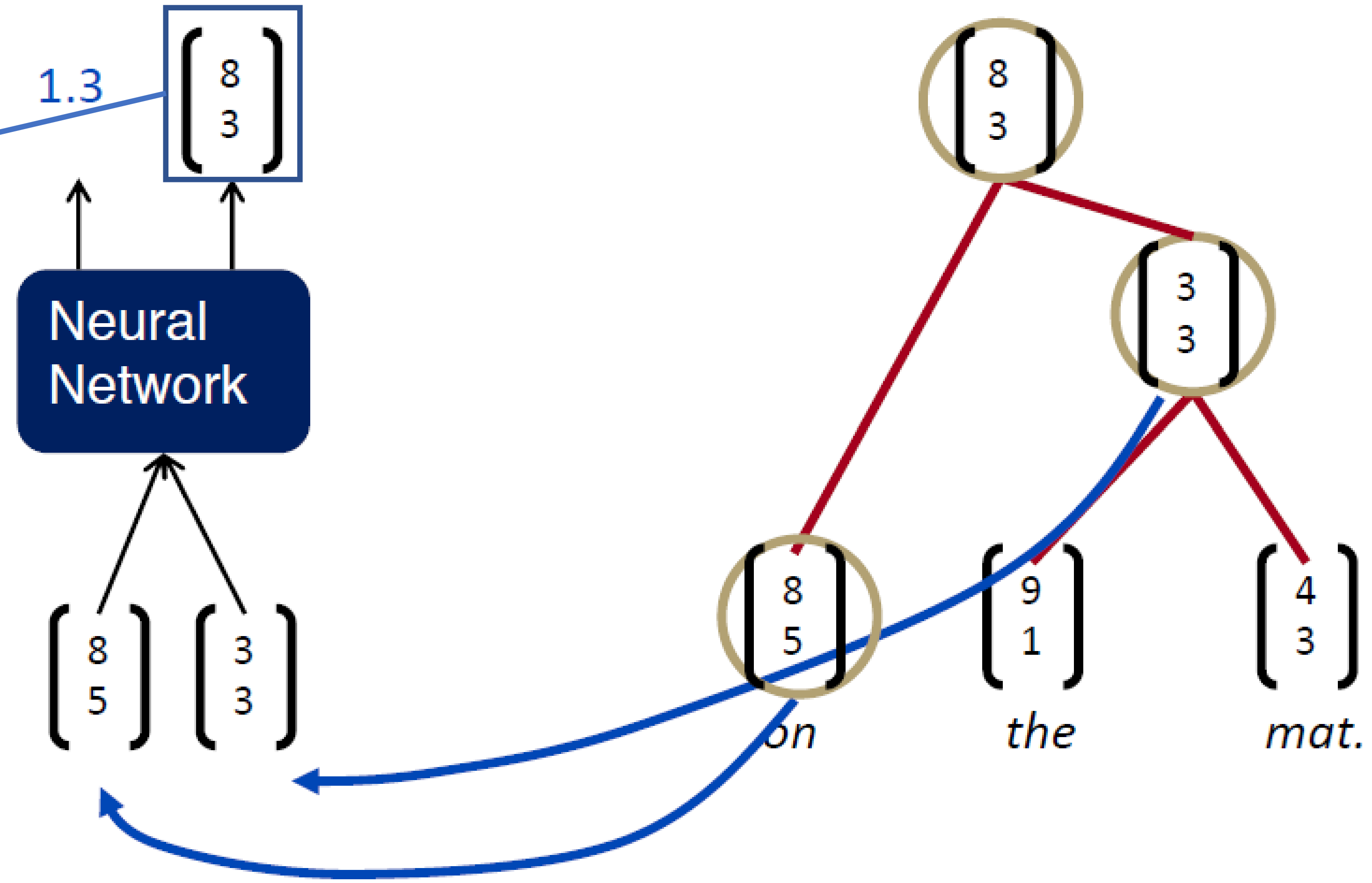
Lecture 18 | Consituency Parsing & Tree RNNs

Parent

$$p = \tanh\left(W\begin{bmatrix} c_1 \\ c_2 \end{bmatrix} + b\right)$$

Concat!

여기서 W 변수는
모든 노드에서 동일!

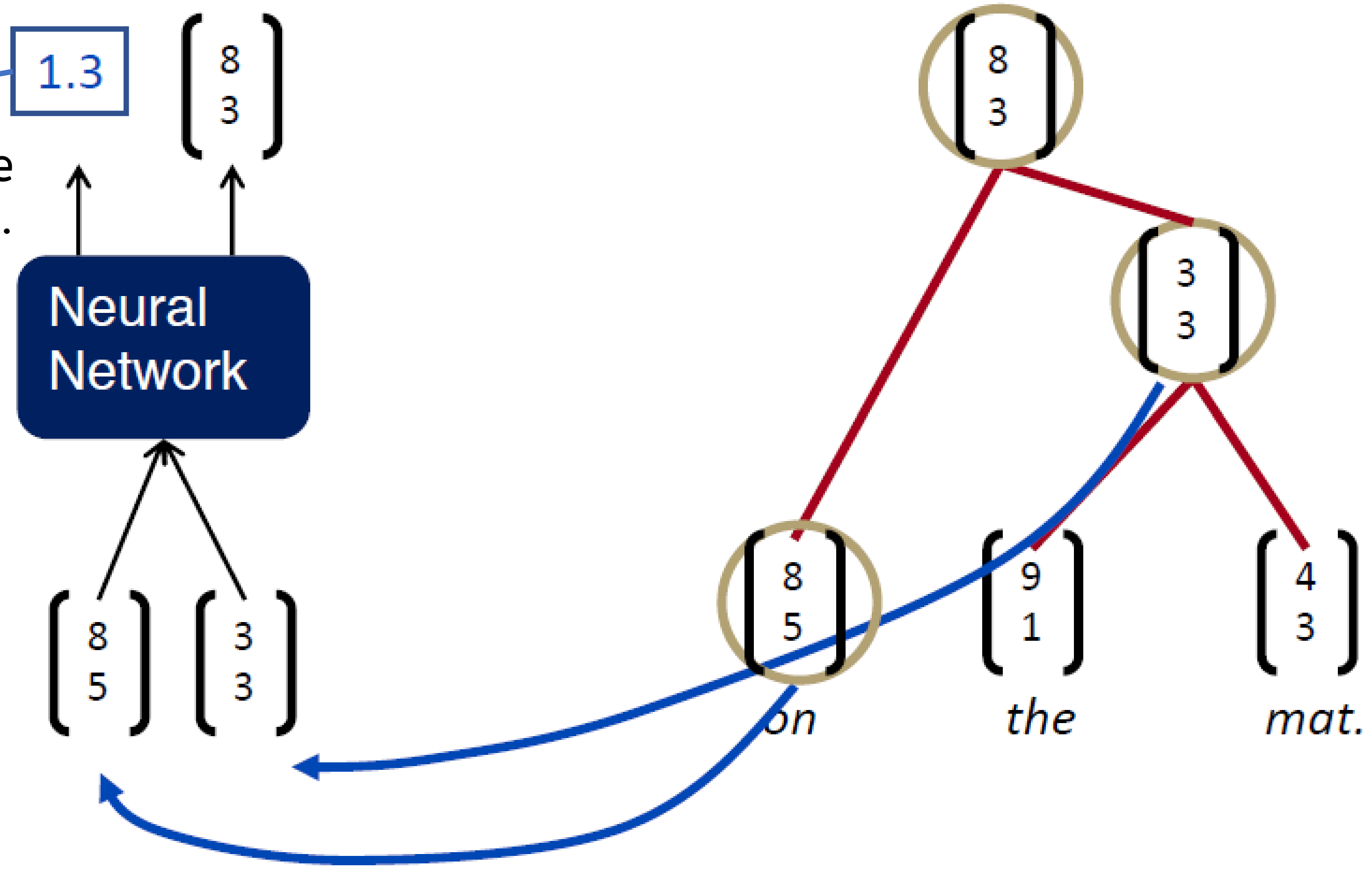


Lecture 18 | Consituency Parsing & Tree RNNs

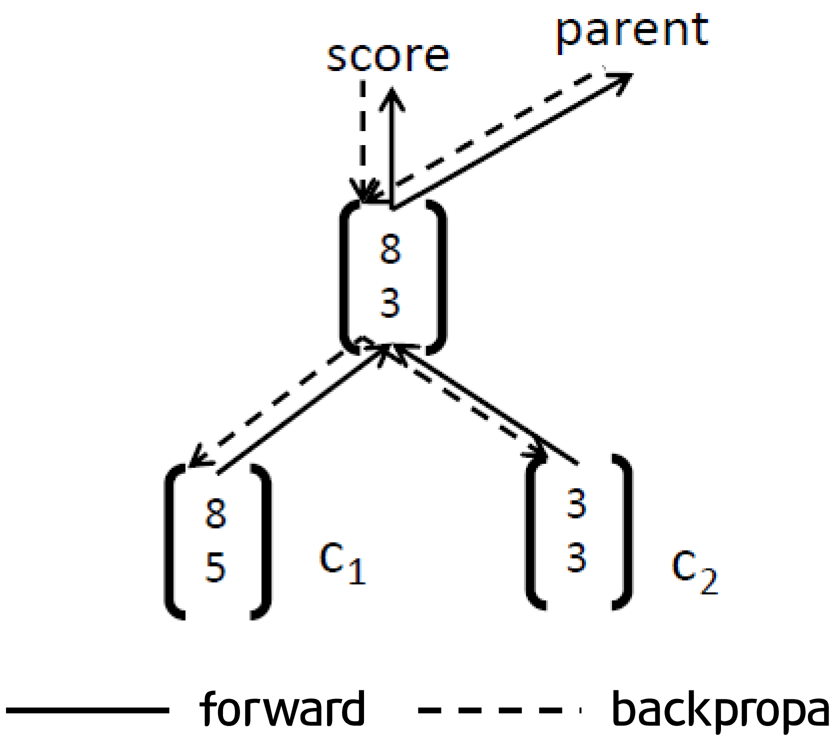
Score of how plausible the new node would be.

$$\text{score} = U^{\mathsf{T}} p$$

$$p = \tanh\left(W \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} + b\right)$$



Simple TreeRNN

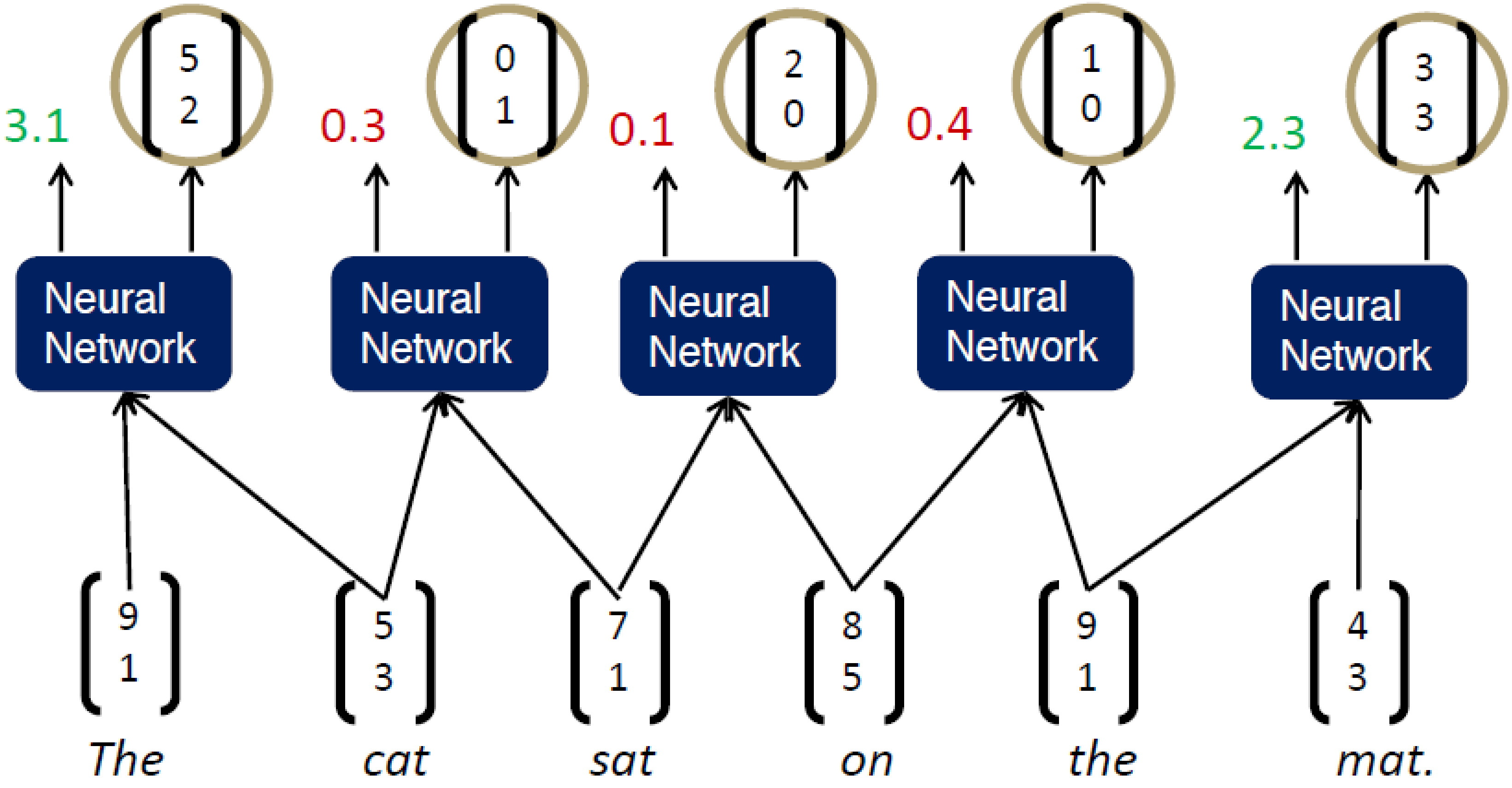


$$p_i = \tanh\left(W \cdot \begin{bmatrix} p_{left} \\ p_{right} \end{bmatrix} + b\right)$$

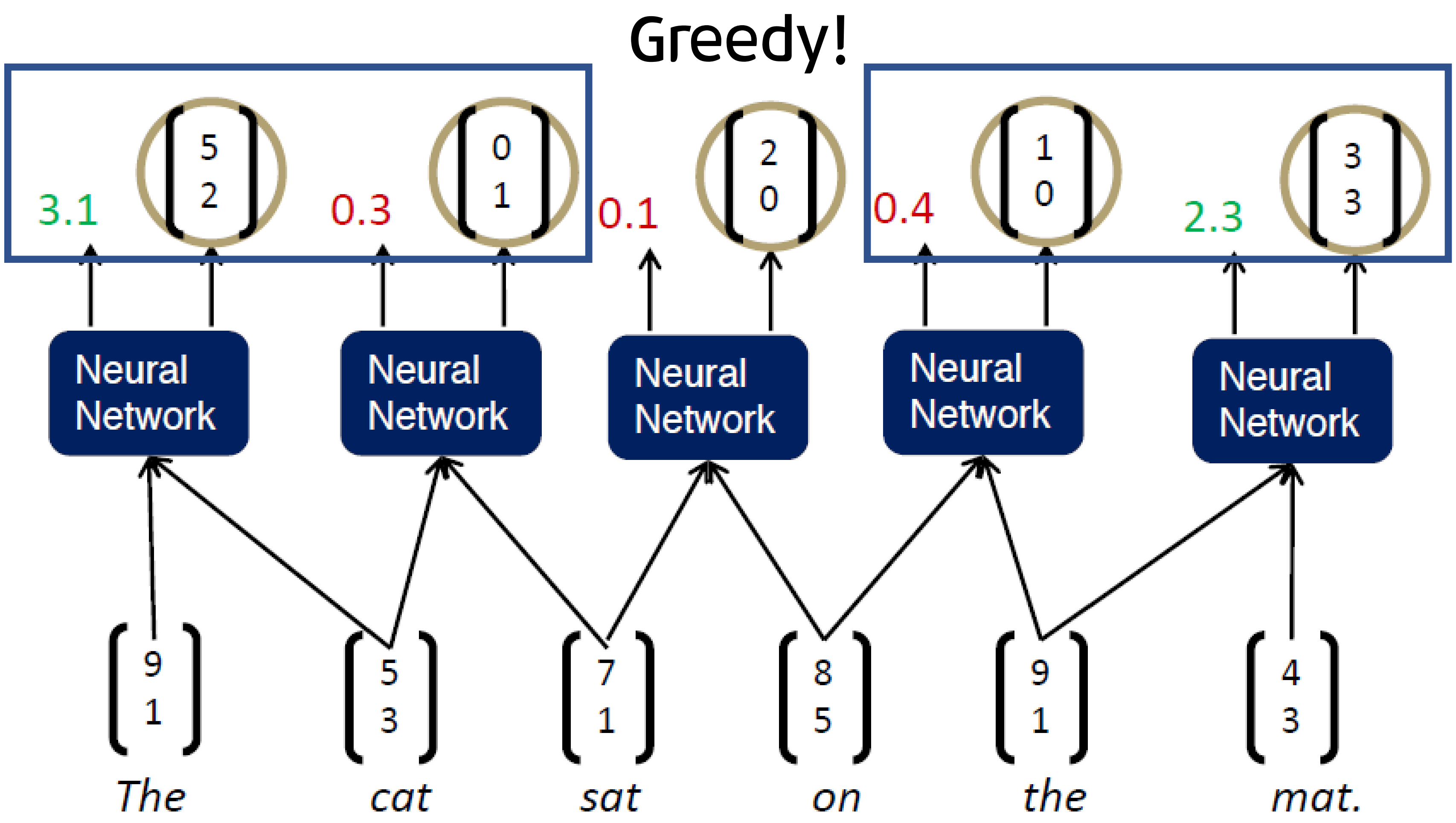
$$W \cdot \begin{bmatrix} p_{left} \\ p_{right} \end{bmatrix} = \begin{bmatrix} w_1 & w_2 \end{bmatrix} \cdot \begin{bmatrix} p_{left} \\ p_{right} \end{bmatrix} = w_1 \times p_{left} + w_2 \times p_{right}$$

$$s_i = W_s p_i + b_s$$

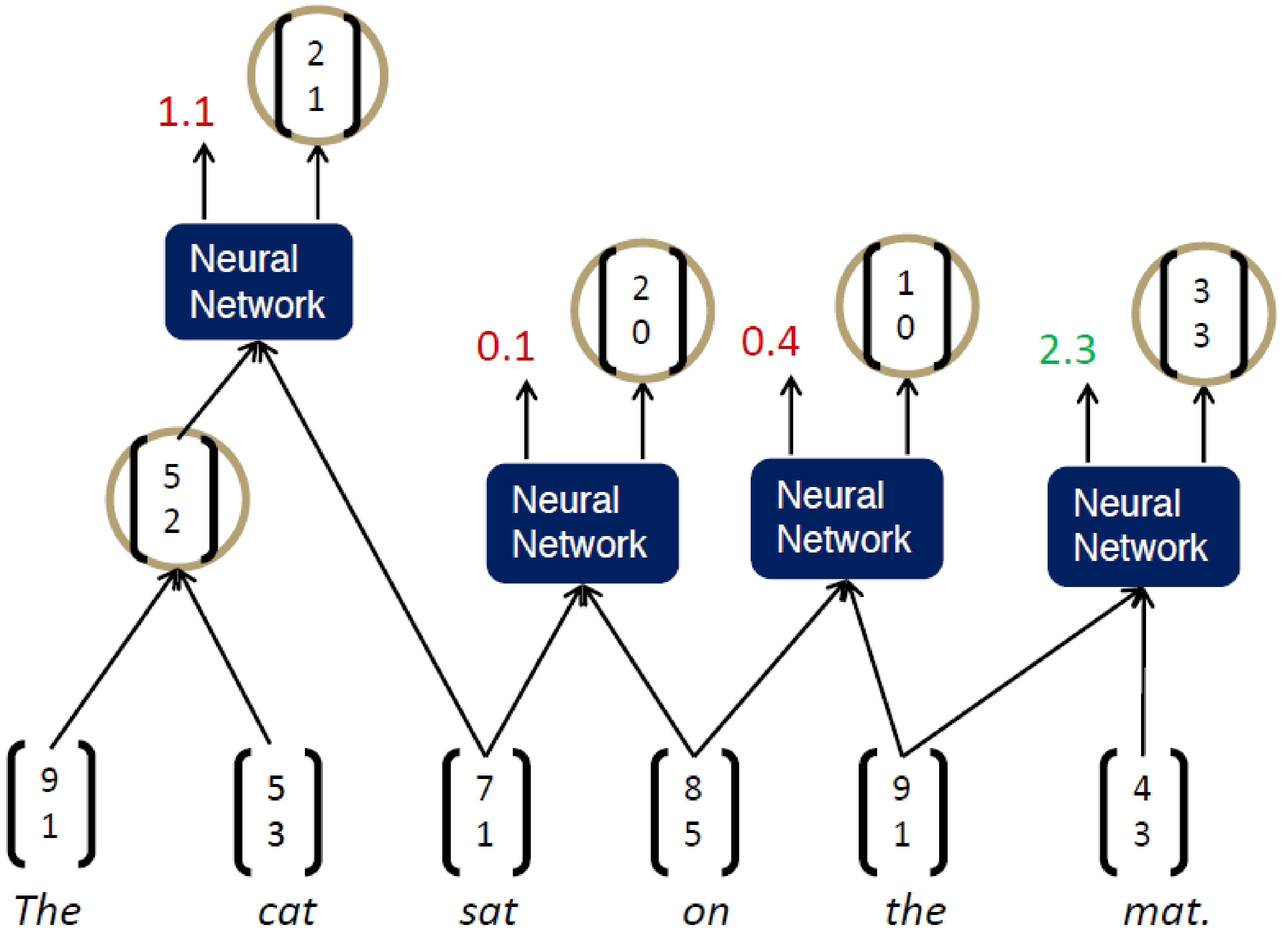
Lecture 18 | Consituency Parsing & Tree RNNs



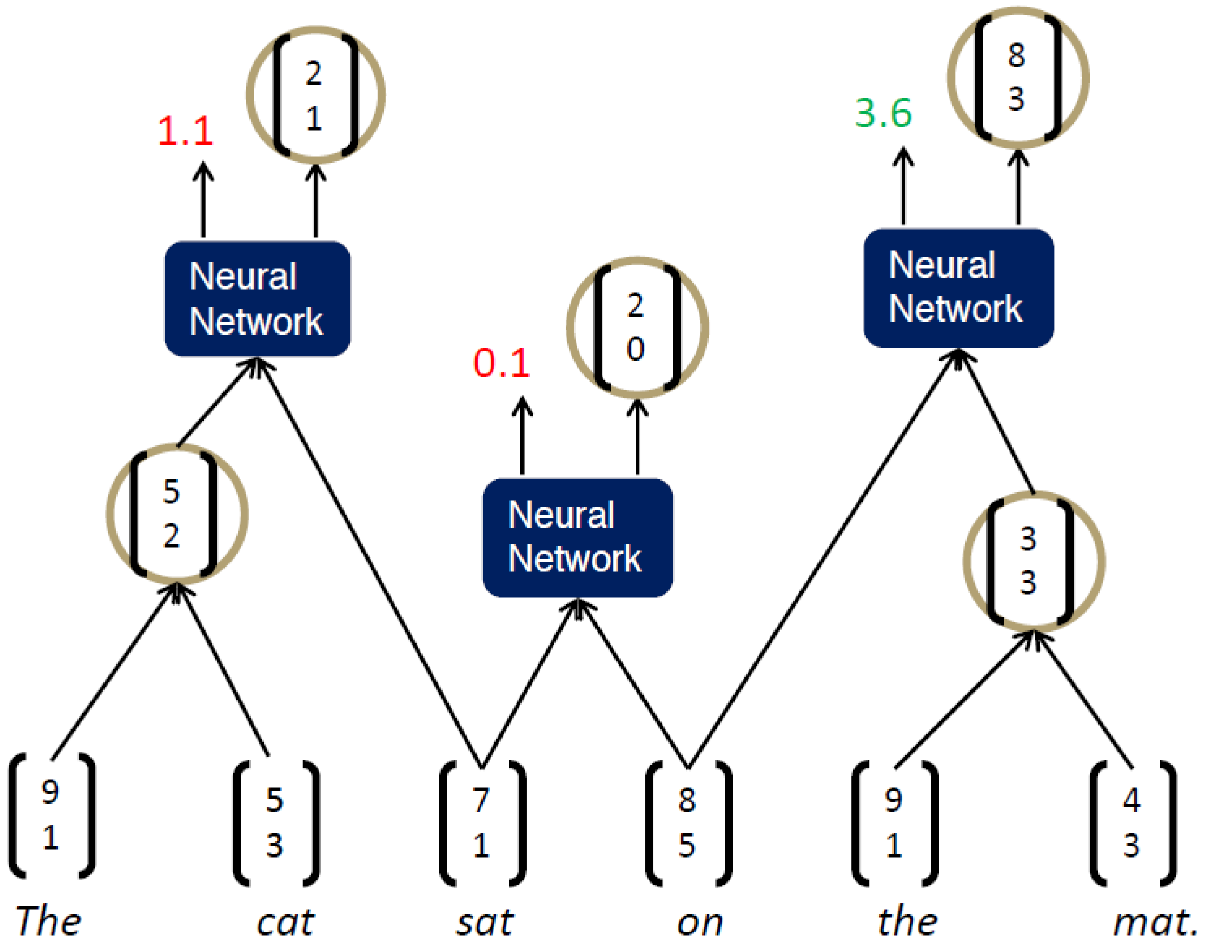
Lecture 18 | Consituency Parsing & Tree RNNs



Lecture 18 | Consituency Parsing & Tree RNNs



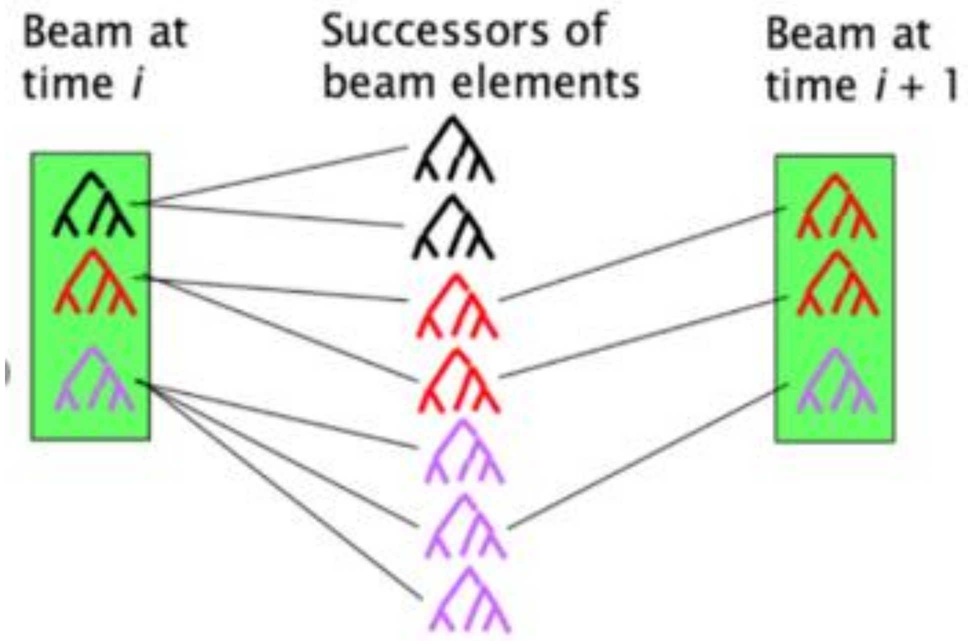
Lecture 18 | Consituency Parsing & Tree RNNs



뜬금 알고리즘 시간!

Dynamic programming Vs greedy algorithm

Beam Search Algorithm



BFS + greedy!

Simple TreeRNN의 한계점

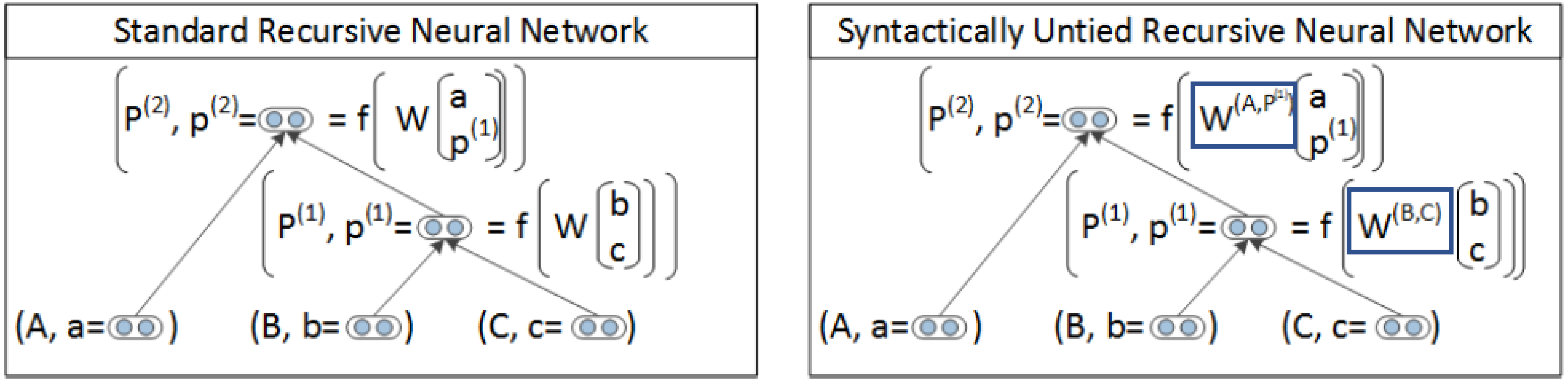
1. 앞서 W가 모든 노드에서 동일하다고 설명. 이러한 Simple TreeRNN은 일부 현상에선 적합할 수 있지만 더 복잡하고, 고차 구성 및 긴 문장에서는 적절치 못하다.

2. 인풋 단어간 실제 상호작용이 없다.

3. 조합 함수가 모든 경우에 대해서 동일하게 작용한다.

Syntactically-United RNN

PCFG 사용. 기능이 다른 표현에 각기 다른 가중치를 적용, case by case. Simple TreeRNN 개선.



Probabilistic Context Free Grammar(PCFG)

각 규칙은 생성 확률을 가지고 있으며 해당 Non-terminal로 부터 생성하는 규칙들의 확률의 합은 1.

S -> NP VP	0.8	
S -> Adj NP VP	0.1	1.0
S -> VP	0.1	

Lecture 18 | Consituency Parsing & Tree RNNs

Grammar	Prob	
S → NP VP	0.8	+ 1.0
S → Aux NP VP	0.1	
S → VP	0.1	
NP → Pronoun	0.2	+ 1.0
NP → Proper-Noun	0.2	
NP → Det Nominal	0.6	
Nominal → Noun	0.3	+ 1.0
Nominal → Nominal Noun	0.2	
Nominal → Nominal PP	0.5	
VP → Verb	0.2	+ 1.0
VP → Verb NP	0.5	
VP → VP PP	0.3	
PP → Prep NP	1.0	

Lexicon

Det → the | a | that | this
0.6 0.2 0.1 0.1

Noun → book | flight | meal | money
0.1 0.5 0.2 0.2

Verb → book | include | prefer
0.5 0.2 0.3

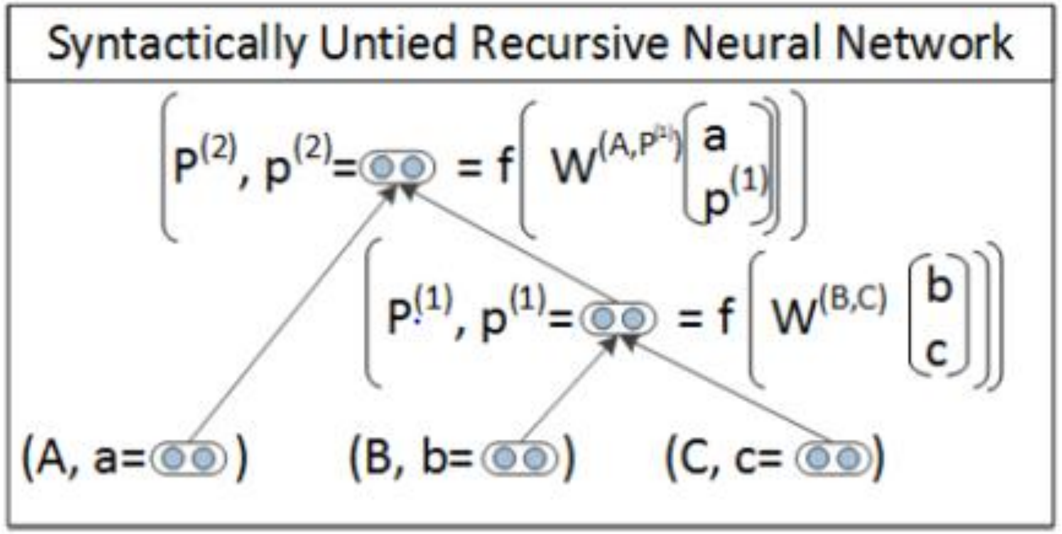
Pronoun → I | he | she | me
0.5 0.1 0.1 0.3

Proper-Noun → Houston | NWA
0.8 0.2

Aux → does
1.0

Prep → from | to | on | near | through
0.25 0.25 0.1 0.2 0.2

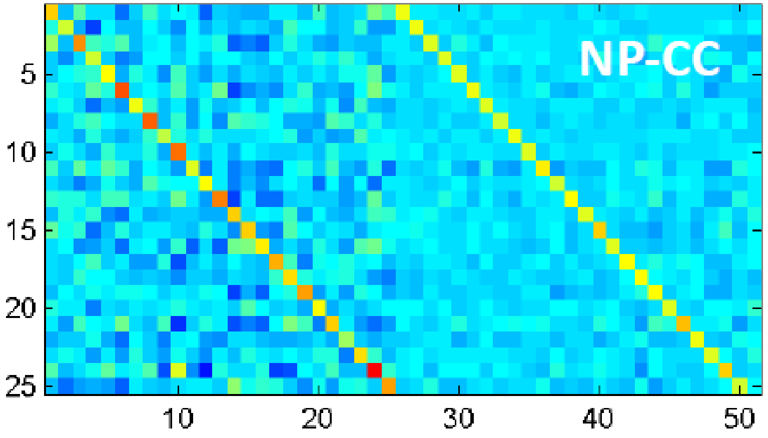
Syntactically-United RNN

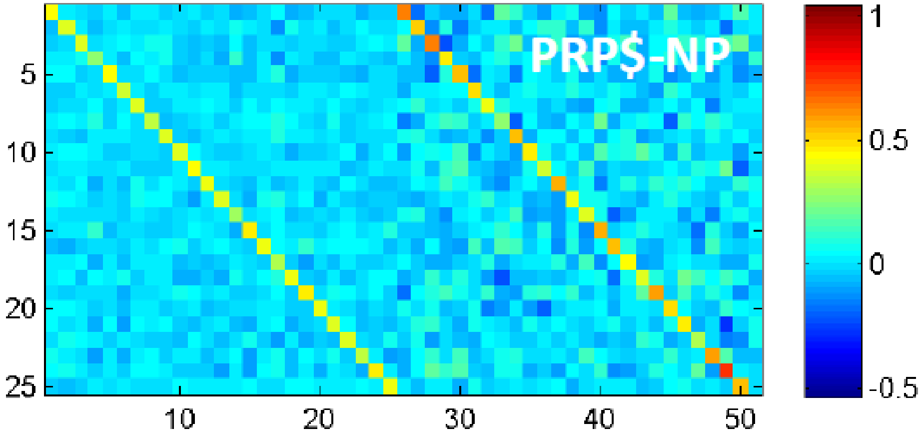


PCFG + TreeRNNs

Lecture 18 | Consituency Parsing & Tree RNNs

Initialization: $W^{(\cdot\cdot)} = 0.5[I_{n\times n} I_{n\times n} 0_{n\times 1}] + \epsilon$

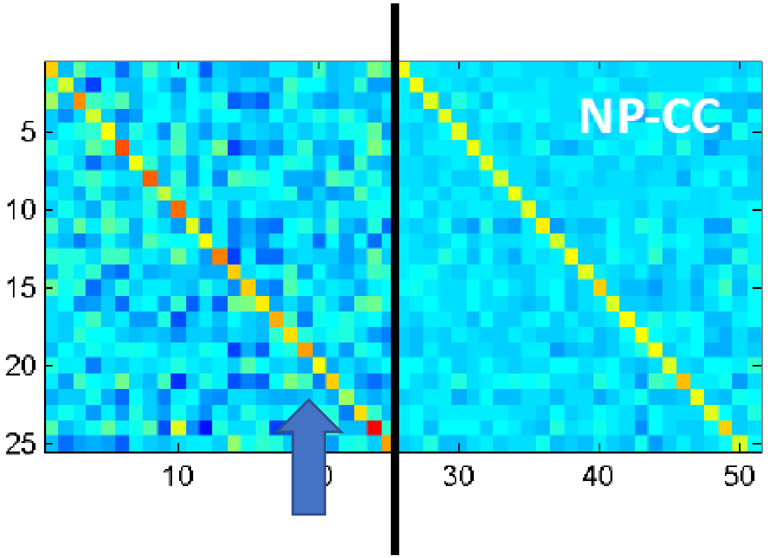


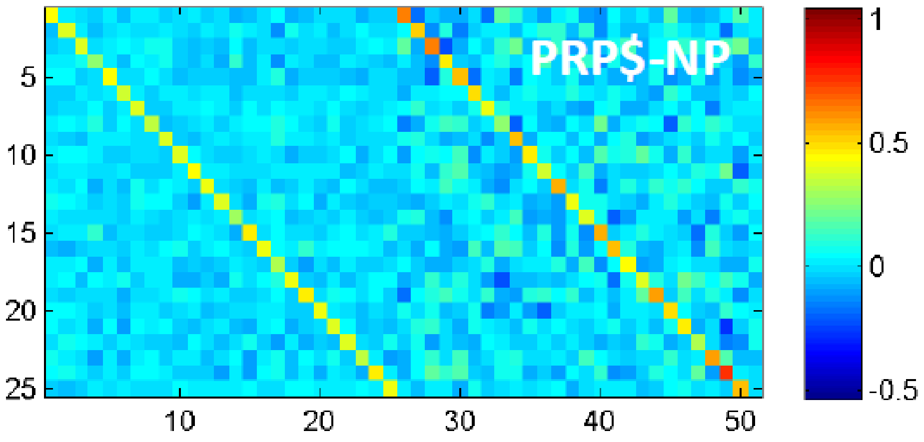


Lecture 18 | Consituency Parsing & Tree RNNs

Initialization: $W^{(\cdot\cdot)} = 0.5[I_{n\times n} I_{n\times n} 0_{n\times 1}] + \epsilon$

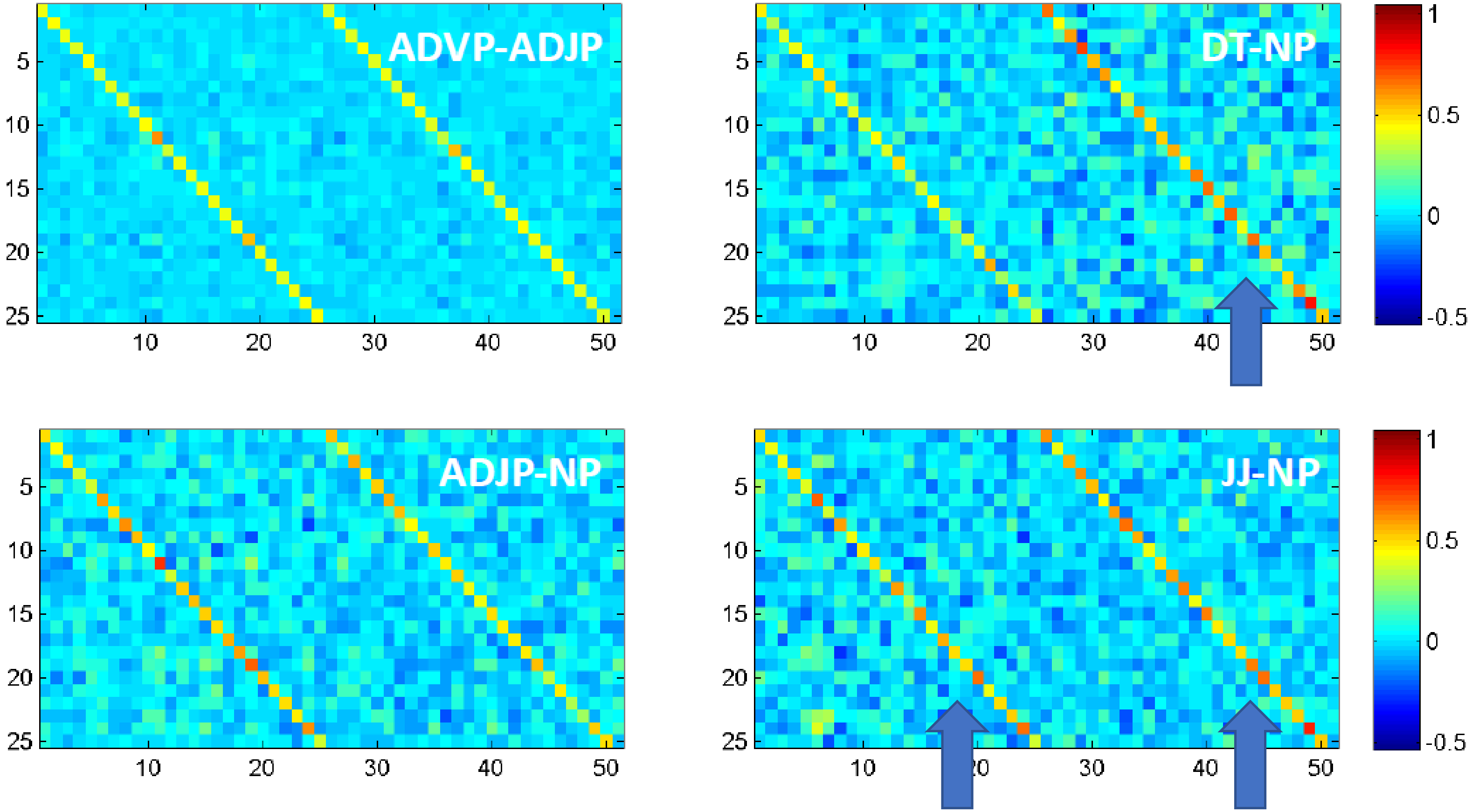
avg Concat bias error



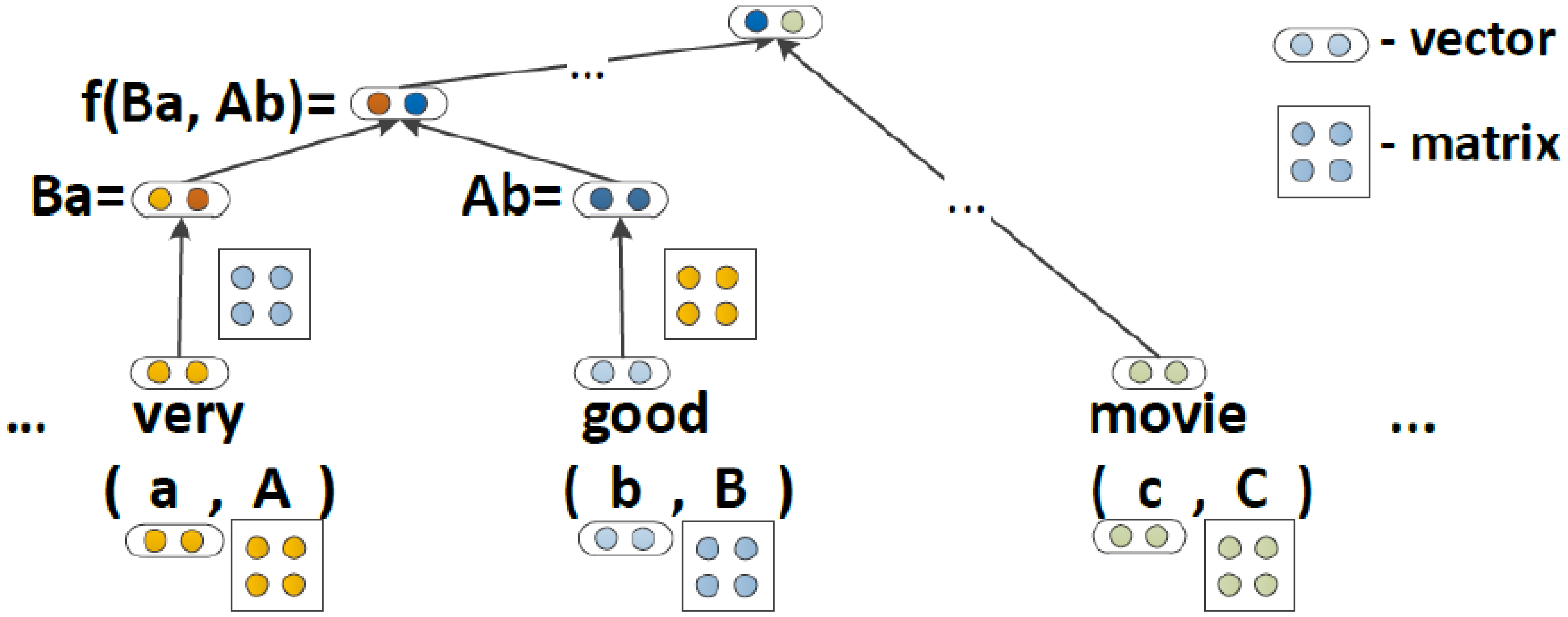


CC 품사보다 NP 품사의 의미가 더 반영되었음! Ex) the cat | and

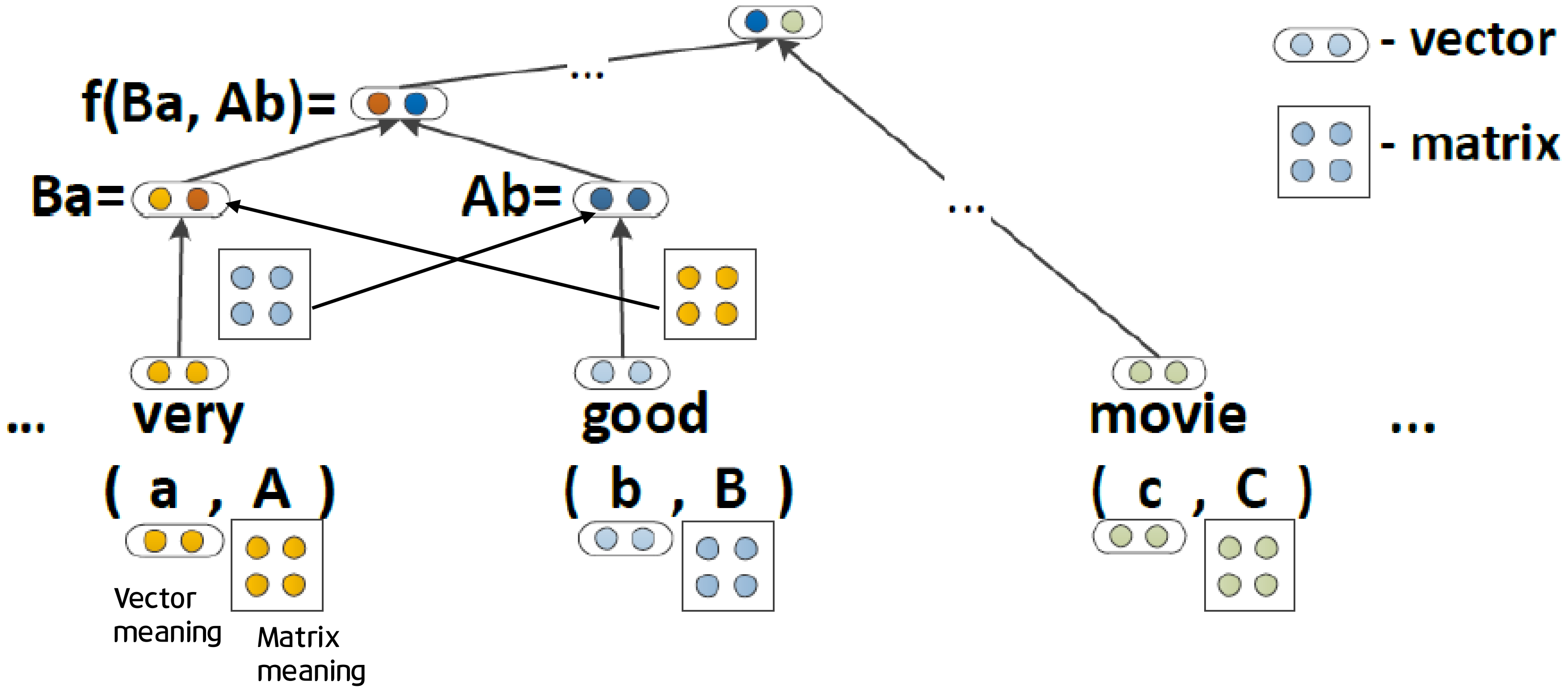
Lecture 18 | Consituency Parsing & Tree RNNs



Matrix-Vector RNN

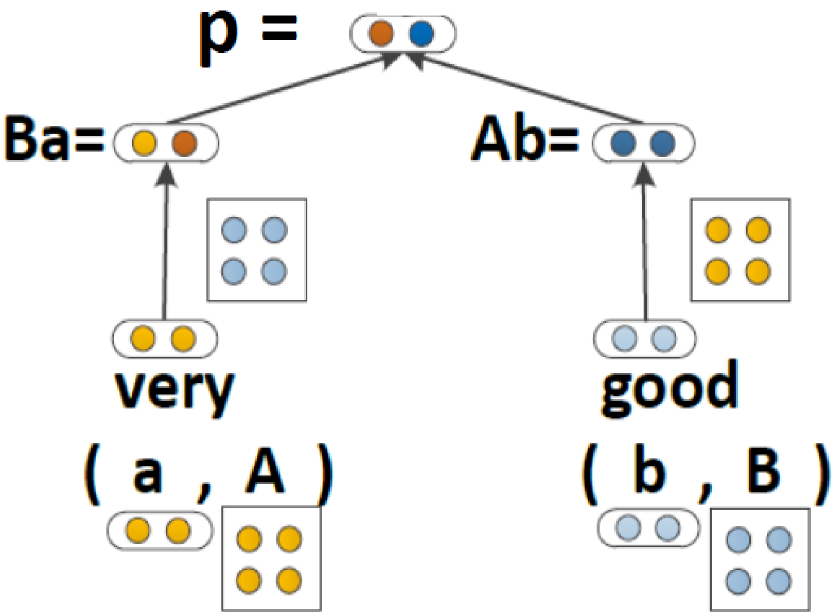


Matrix-Vector RNN



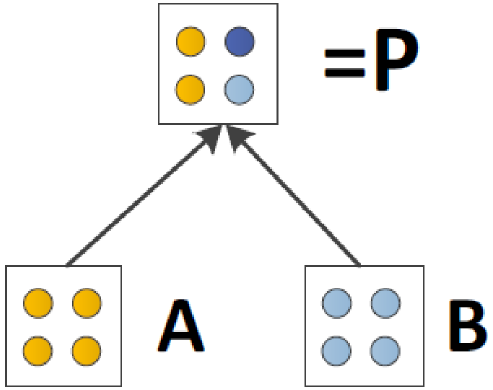
Matrix-Vector RNN

$$p = f\left(W \begin{bmatrix} Ba \\ Ab \end{bmatrix}\right)$$

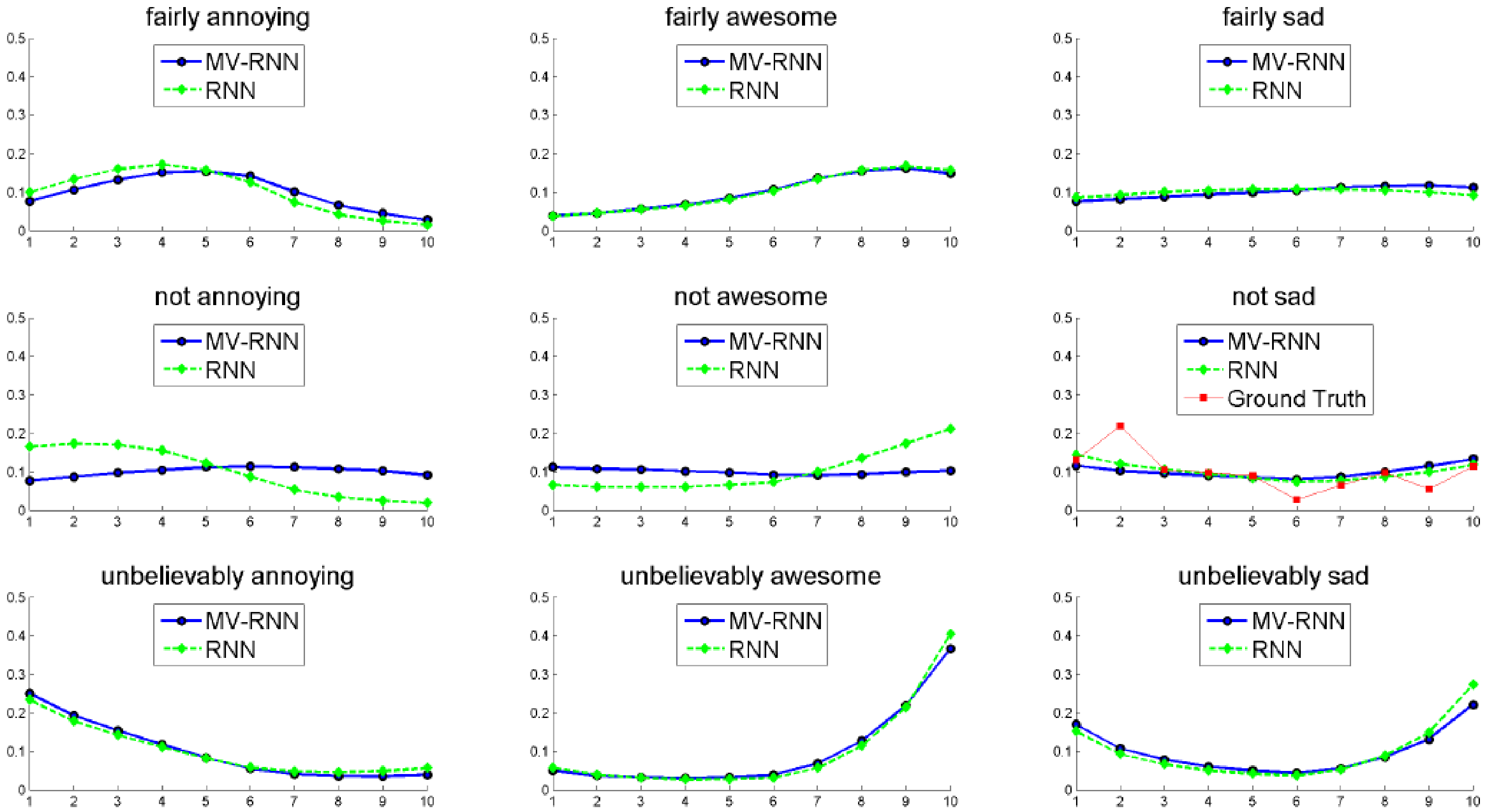


$$P = g(A, B) = W_M \begin{bmatrix} A \\ B \end{bmatrix}$$

$$W_M \in \mathbb{R}^{n \times 2n}$$



Lecture 18 | Consituency Parsing & Tree RNNs



Sentiment detection

사실 감정분석을 하는건 어려운일은 아님.

Bag of Words를 통해 임베딩하여 문장을 분류하여도 90% 정도의 성능을 보인다고함.

Lecture 18 | Consituency Parsing & Tree RNNs

With this cast, and this subject matter, the movie

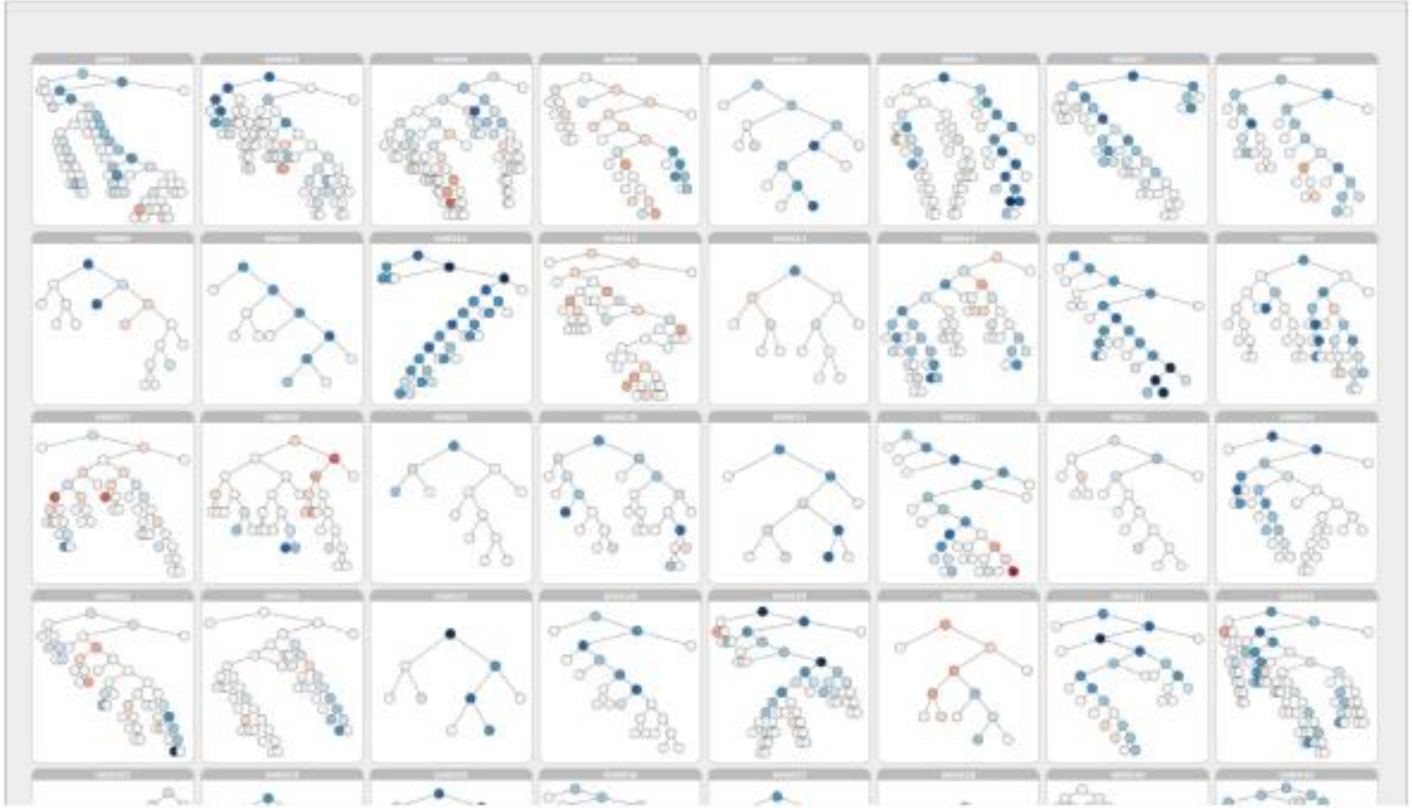
should have been funnier and more entertaining

슈드 해브 피피..

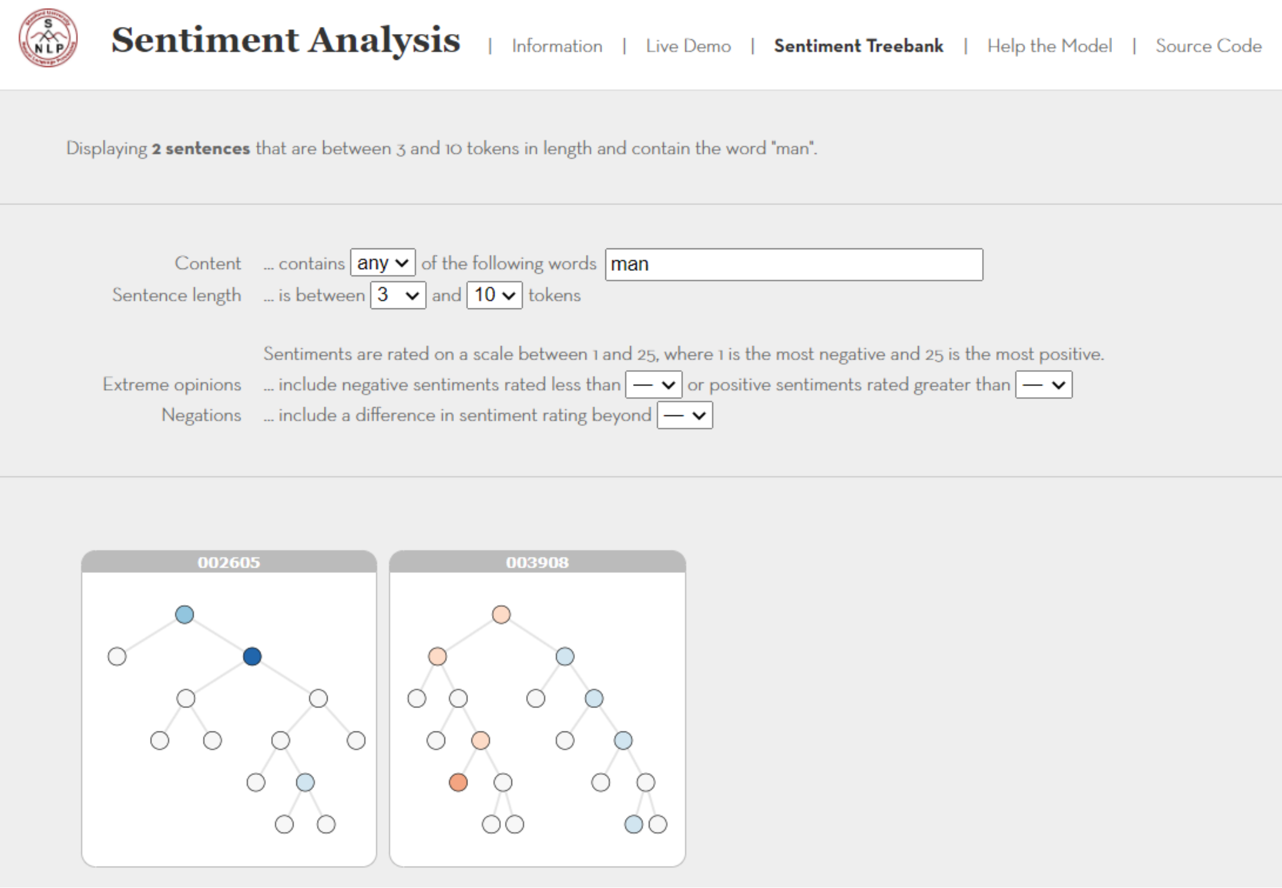
후회 + 긍정 -> 부정, meaning composition을 위해선 또 다른 무언가 필요!

Lecture 18 | Consituency Parsing & Tree RNNs

Stanford Sentiment Treebank

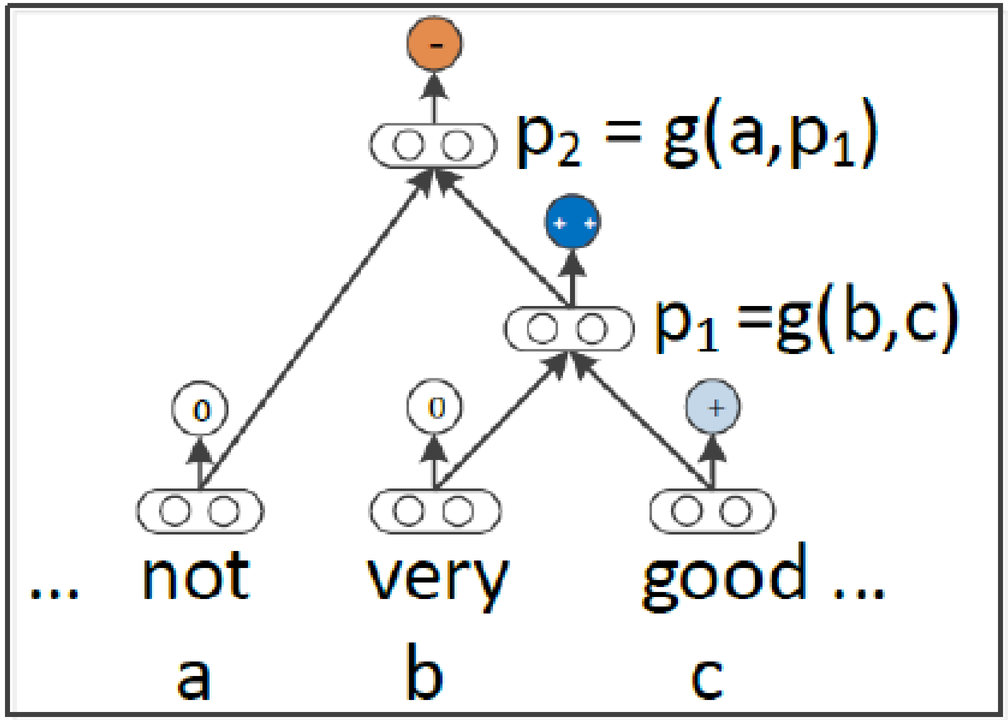


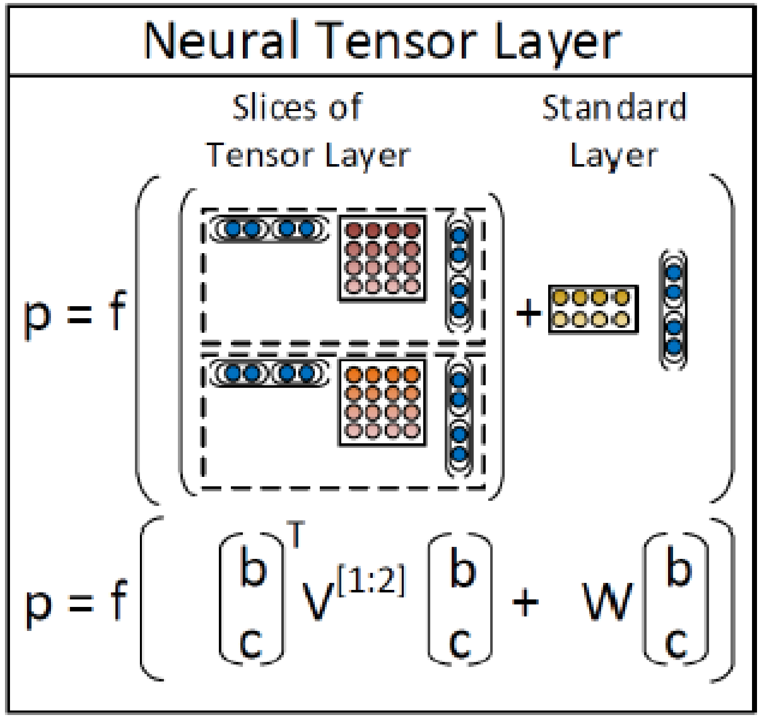
Lecture 18 | Consituency Parsing & Tree RNNs



https://nlp.stanford.edu/sentiment/treebank.html

Recursive Neural Tensor Network





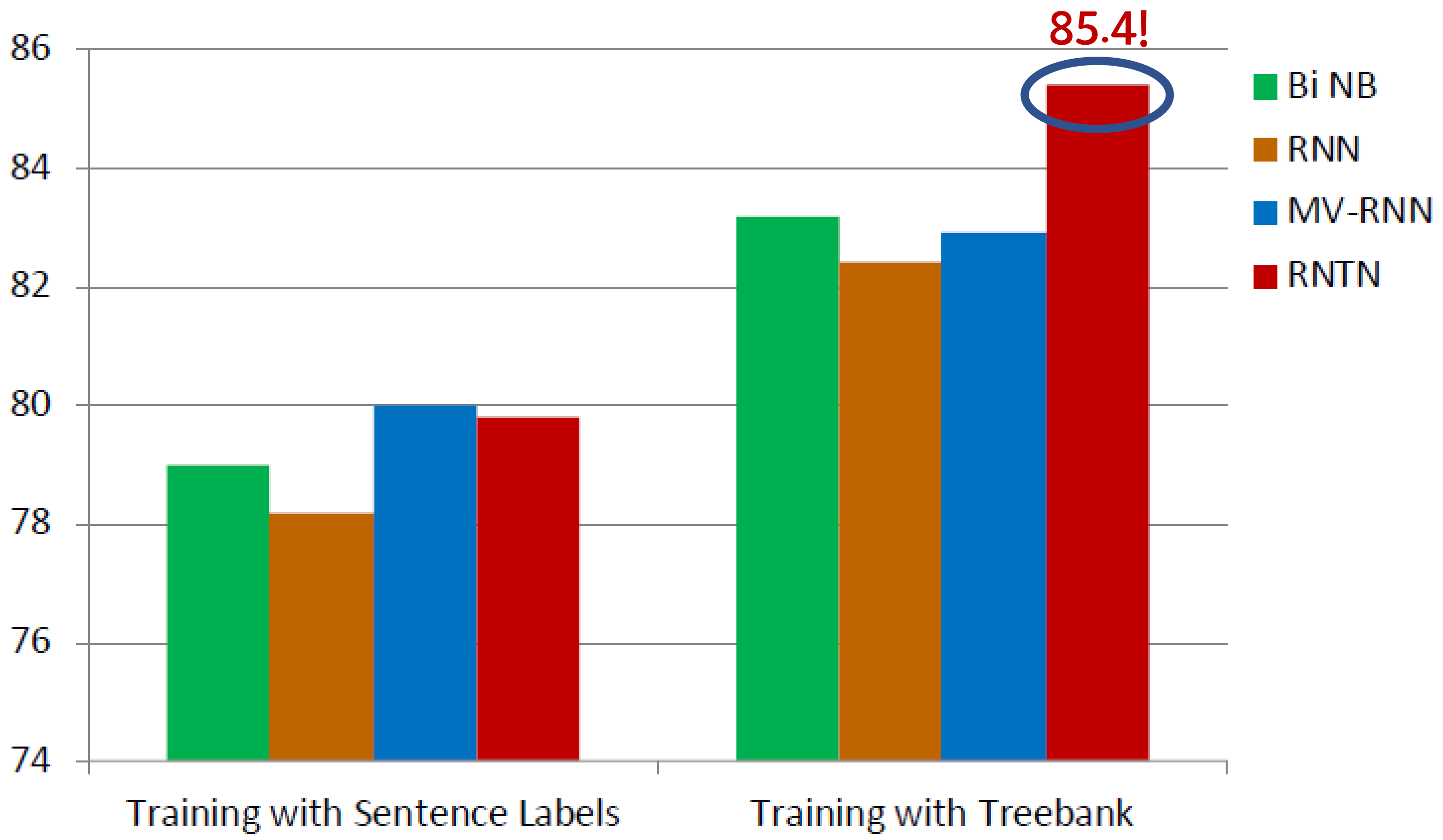
MV-RNN보다 더 적은 파라미터를 가짐.

MV-RNN은 모든 결합에 대해 각 노드마다 행렬을 생성하기 때문에 cost가 높음.

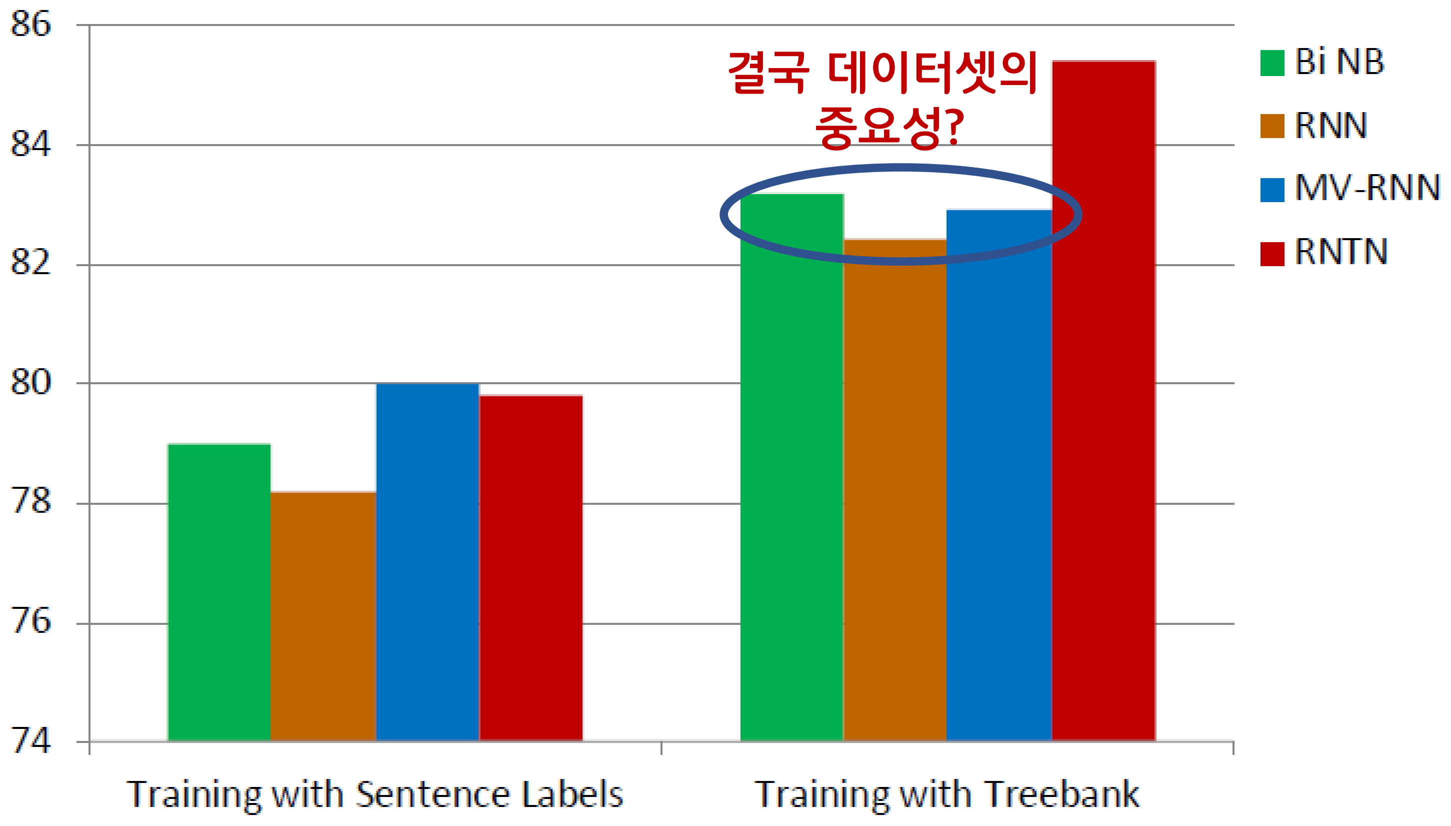
Lecture 18 | Consituency Parsing & Tree RNNs

구분	simple Recursive Neural Network	MV-RNN	RNTN
Word Vector	$d \times 1$	$d \times 1$	$d \times 1$
Softmax W_S	$C \times d$	$C \times d$	$C \times d$
Compostion Weights	$W : d \times 2d$	$W : d \times 2d,$ $W_M : d \times 2d,$ $A, B, C \ldots : d \times d \times \lvert v \rvert$	$W : d \times 2d,$ $V^{[1:d]} : 2d \times 2d \times d$

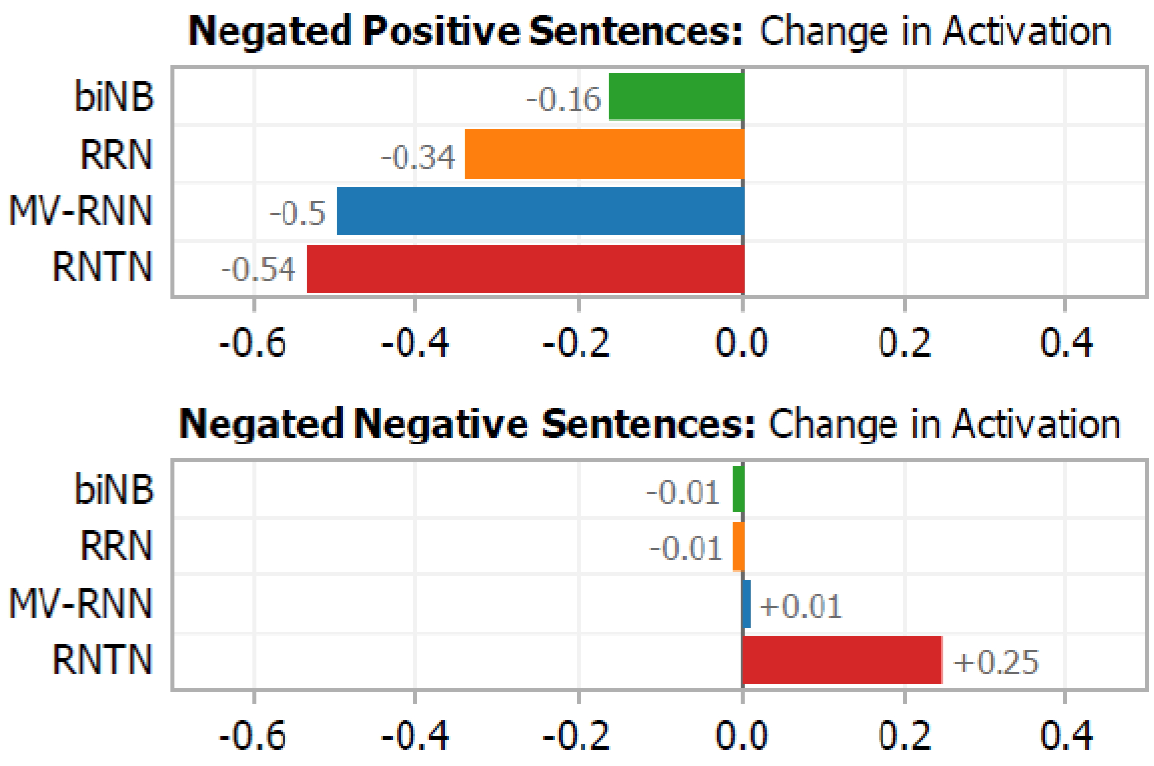
Positive/Negative Results on Treebank



Positive/Negative Results on Treebank



Lecture 18 | Consituency Parsing & Tree RNNs



Lecture 18 | Consituency Parsing & Tree RNNs

한계점

1. Gpu 연산이 힘듦. 병렬적으로 이루어 져야하는데 모든 task 마다 연산 구조가 동일하지 않음. Tree 모양이 다름.

2. 데이터셋 구축이 어려움. 라벨링 ㅠ

Lecture 18 | Consituency Parsing & Tree RNNs

Tree-to-tree Neural Networks for Program Translation

Xinyun Chen
UC Berkeley
xinyun.chen@berkeley.edu

Chang Liu
UC Berkeley
liuchang2005acm@gmail.com

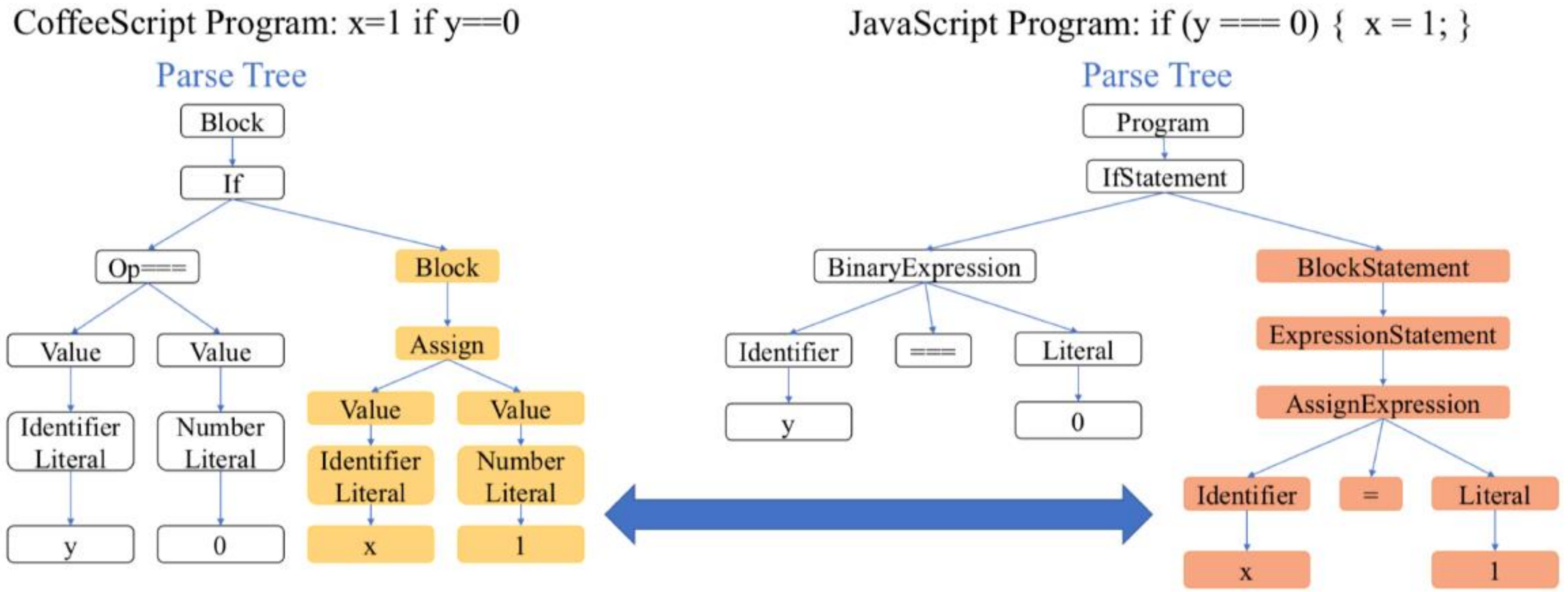
Dawn Song
UC Berkeley
dawnsong@cs.berkeley.edu

Abstract

Program translation is an important tool to migrate legacy code in one language into an ecosystem built in a different language. In this work, we are the first to employ deep neural networks toward tackling this problem. We observe that program translation is a modular procedure, in which a sub-tree of the source tree is translated into the corresponding target sub-tree at each step. To capture this intuition, we design a tree-to-tree neural network to translate a source tree into a target one. Meanwhile, we develop an attention mechanism for the tree-to-tree model, so that when the decoder expands one non-terminal in the target tree, the attention mechanism locates the corresponding sub-tree in the source tree to guide the expansion of the decoder. We evaluate the program translation capability of our tree-to-tree model against several state-of-the-art approaches. Compared against other neural translation models, we observe that our approach is consistently better than the baselines with a margin of up to 15 points. Further, our approach can improve the previous state-of-the-art program translation approaches by a margin of 20 points on the translation of real-world projects.

https://papers.nips.cc/paper/2018/file/d759175de8ea5b1d9a2660e45554894f-Paper.pdf

Tree-to-tree Nerual Networks for Program Translation



References

1. CS224n lecture 18: Consituency Parsing, Tree RNNs

2. https://ratsgo.github.io/deep%20learning/2017/04/03/recursive/

3. https://ratsgo.github.io/deep%20learning/2017/06/24/RNTN/

4. https://www.youtube.com/watch?v=TcNvkPoaXas&ab_channel=KoreaUnivDSBA