Constituency Parsing, TreeRNNs

CS224n Natural Language Processing with Deep Learning
- Lecture 18 -

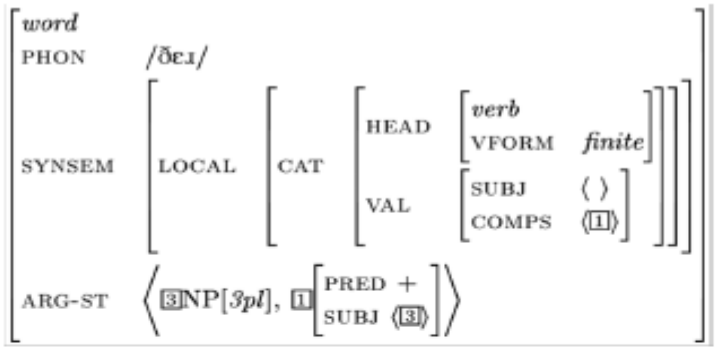
UOS STAT NLP Study
Hyehyeon Moon

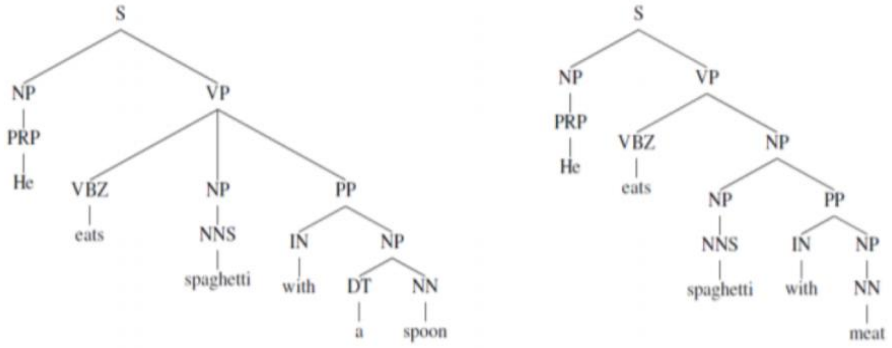
2021.02.20

Index

Motivation

Constituency Parsing





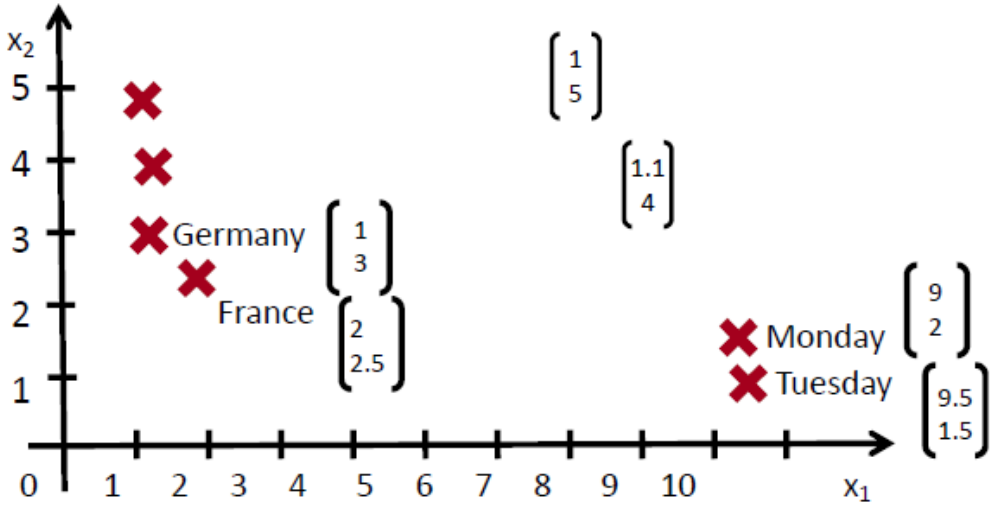
Compositionality

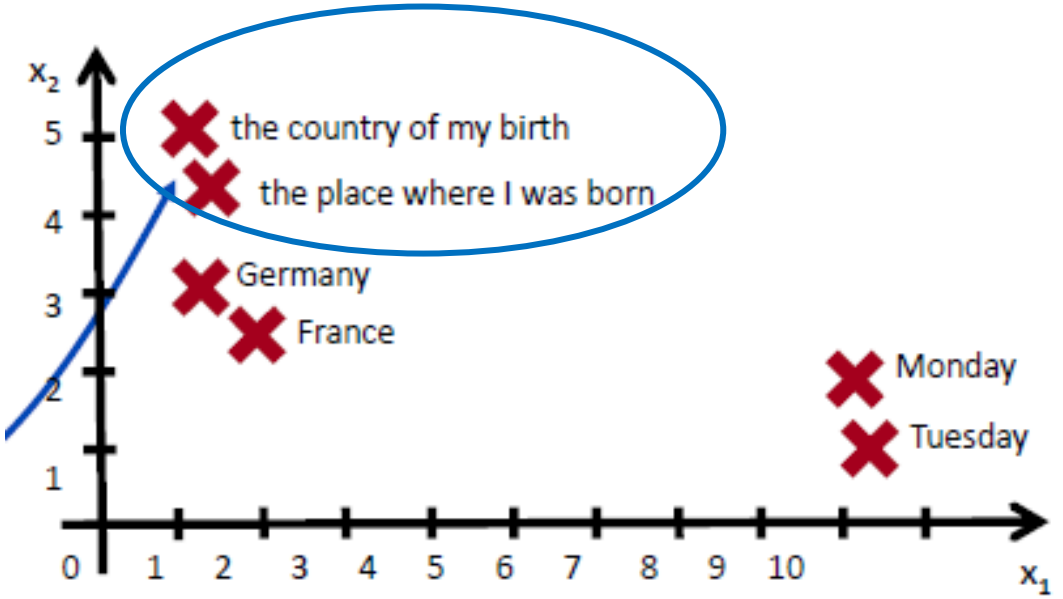
- *The **snowboarder** is leaping over a mogul*
- *A **person on a snowboard** jumps into the air*

- 'A person on a snowboard'의미와 'snowboarder'의 의미는 동일합니다.
- 이렇게 'A person on a snowboard'에서 다섯 단어의 조합으로 snowboarder를 표현할 수 있게 됩니다.
- 즉슨 단어의 조합으로 부터 문장의 의미를 파악할 수 있다는 말로 해석됩니다.

Motivation

Meaning(vector) of a sentence



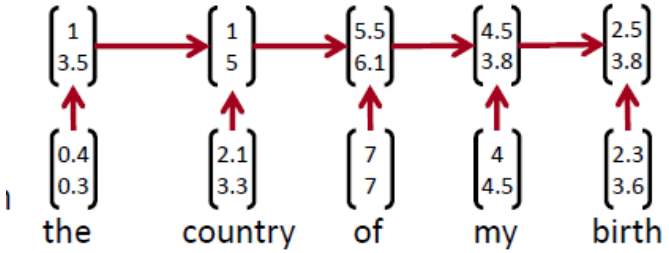


Use principle of compositionality
The meaning (vector) of a sentence is determined by (1) the meanings of its words and (2) the rules that combine them.

Simple Tree RNN

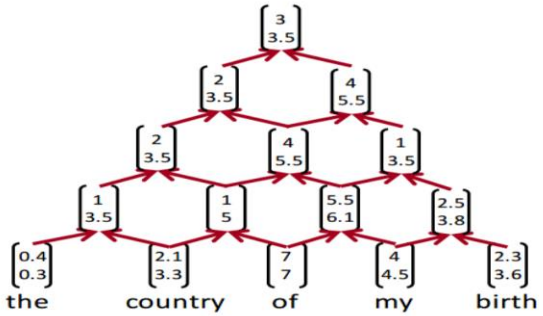
Recurrent NN vs Convolution NN vs Recursive NN

Recurrent



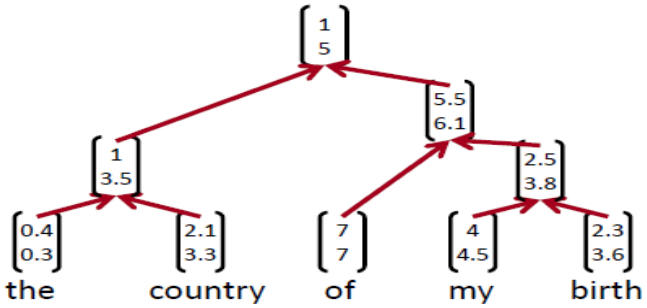
- 입력값을 순서대로 받아 하나씩 순차적으로 처리하는 네트워크 구조
- Cannot capture phrases without prefix context
- Often capeture too much of last words in final vector

Convolution



- 필터의 크기가 단어 2개로 세팅되어 있는데, 이 필터가 한칸씩 슬라이딩하면서 문장을 단어 두개씩 읽어들여 분석하는 구조
- 입력값이 한번에 주어지고 필터가 슬라이딩하면서 문장의 지역적인 정보를 반영한다는 점

Recursive



- 입력값으로 주어지는 몇 개 단어를 묶어서 분석
- 일부 정보는 스킵한다는 점

Simple Tree RNN

Recursive NN for Structure Prediction

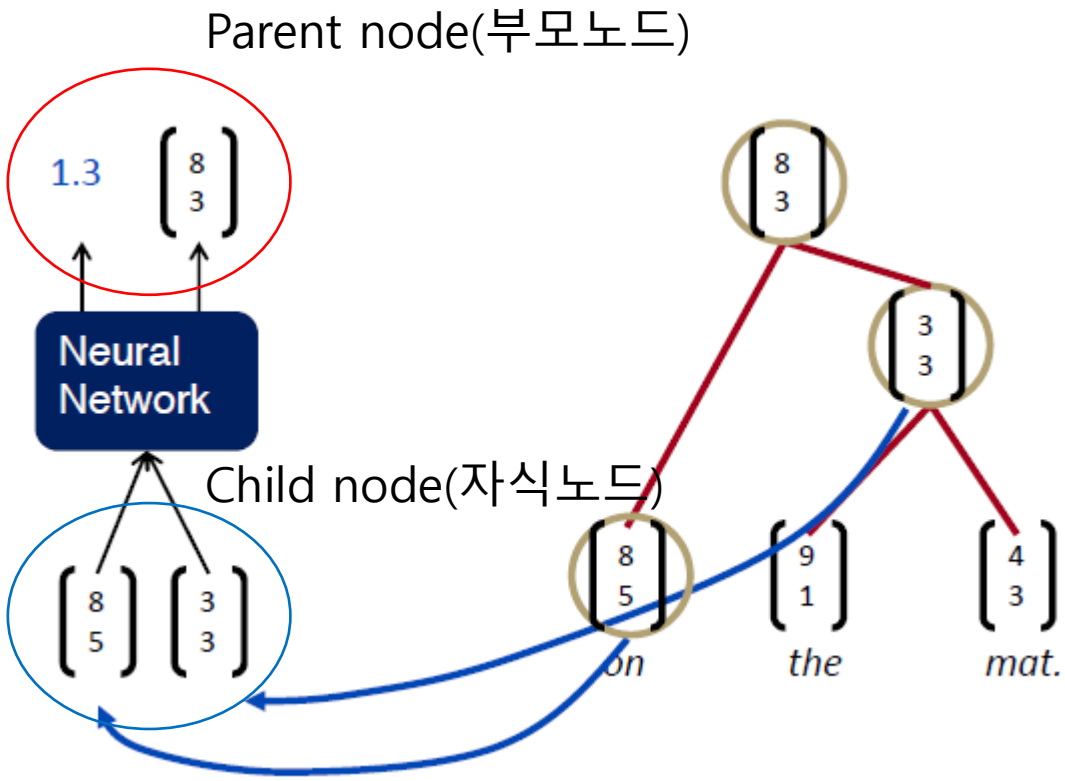
- Input

two candidate children's representations

- Outputs

Semantic representation

Score of how plausible the new node would be

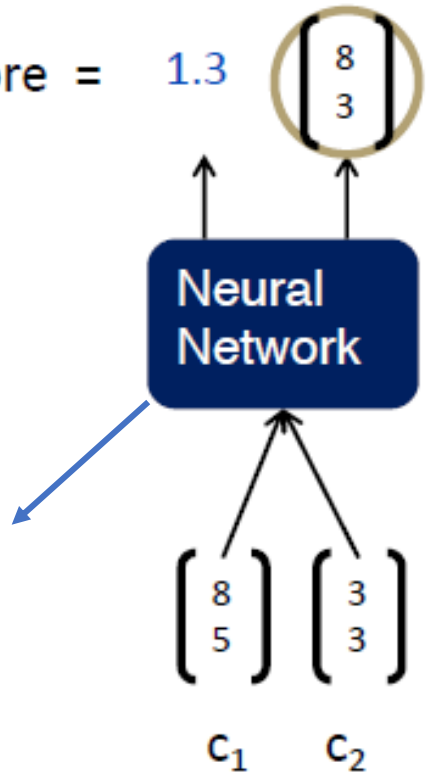


- 만약 파싱을 위한 RNN이라면 이 스코어는 합쳐진 단어들이 얼마나 말이 되는지를 나타내는 점수가 될 것이고,
- 감성분석을 위한 RNN이라면 해당 점수는 긍/부정 극성을 나타내는 지표가 됩니다.

Simple Tree RNN

Recursive NN Definition

$$s_i = W_s p_i + b_s$$



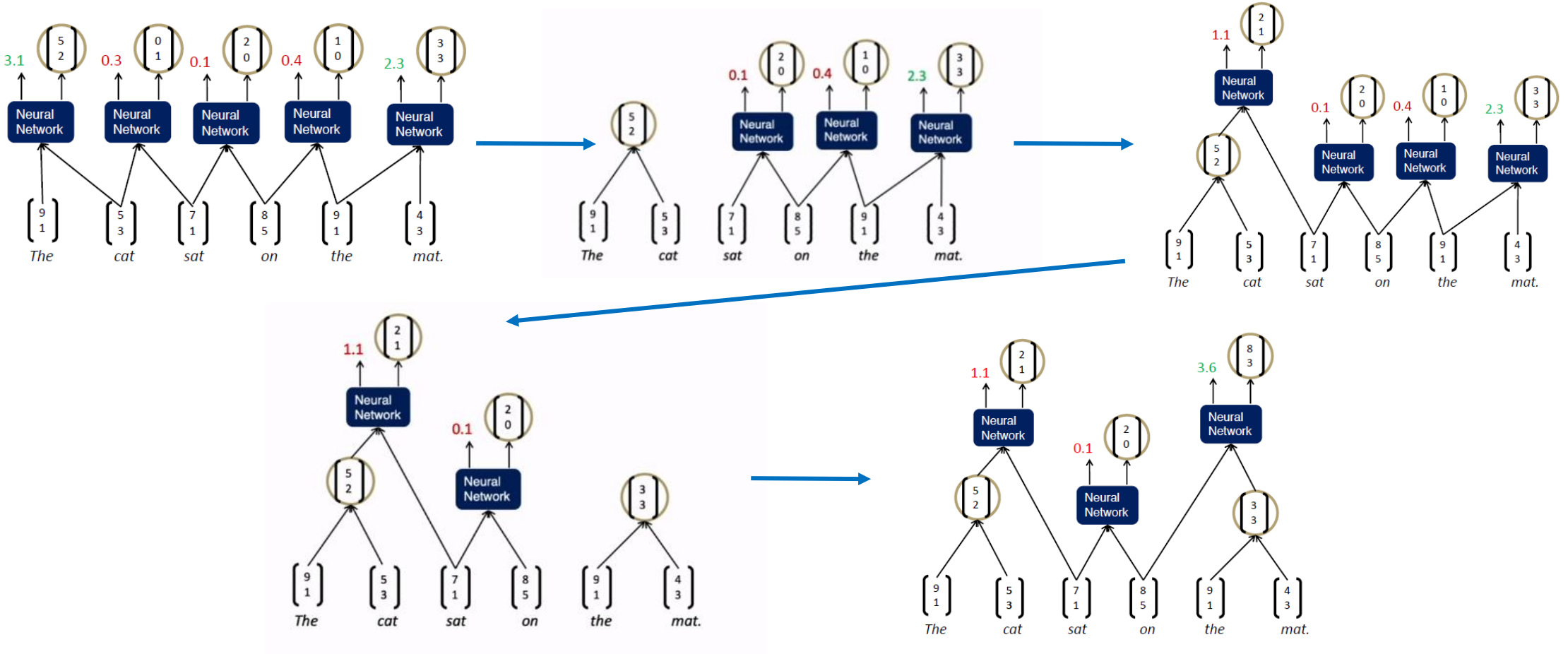
$$p_i = \tanh\left(W \cdot \begin{bmatrix} p_{left} \\ p_{right} \end{bmatrix} + b\right)$$

$$p_i \in R^d, \quad b \in R^d, \quad W \in R^{d \times 2d}$$

Same W parameters at all nodes of the tree

Simple Tree RNN

Parsing a sentence with an RNN(greedily)



Simple Tree RNN

Details

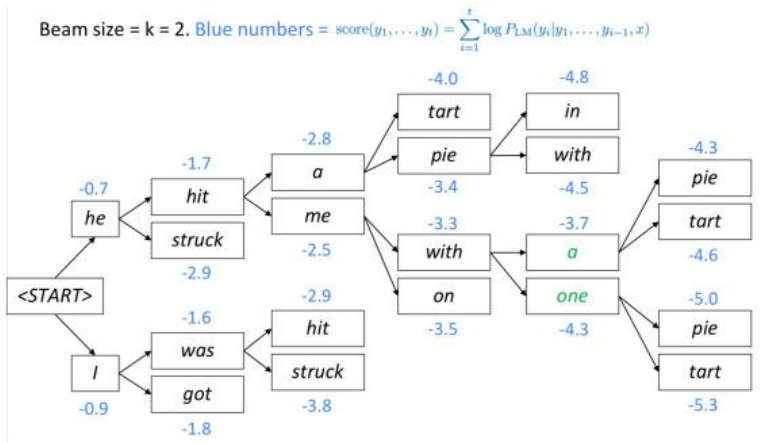
- The score of a tree is computed by the sum of the parsing decision scores at each node.

$$s(x, y) = \sum_{n \in nodes(y)} s_n$$

- Use max-margin objective

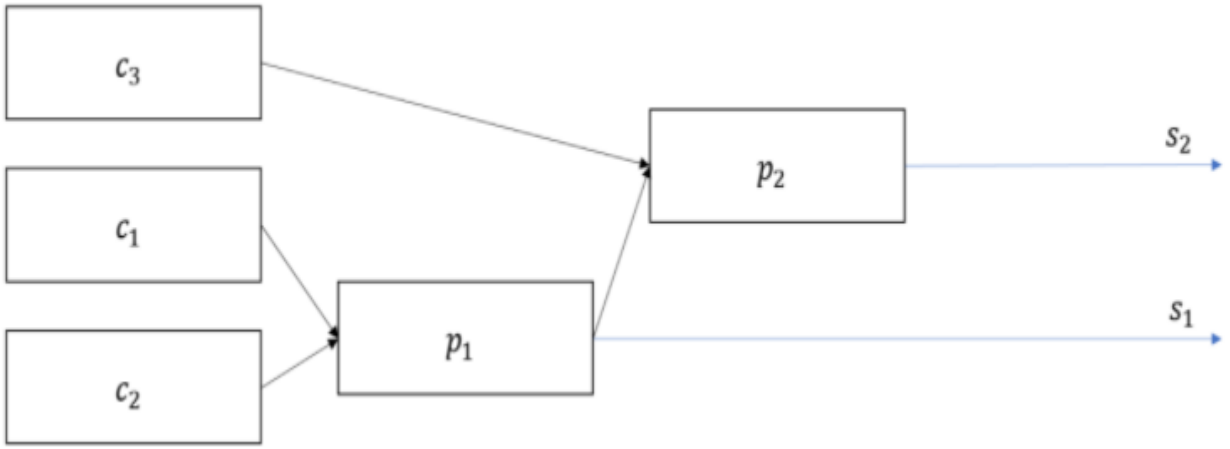
$$J = \sum_i s(x_i, y_i) - \max_{y \in A(x_i)} \left(s(x_i, y) + \Delta(y, y_i)\right)$$

- We can use Beam search with chart instead of greedy algorithm.



Simple Tree RNN

Forward pass



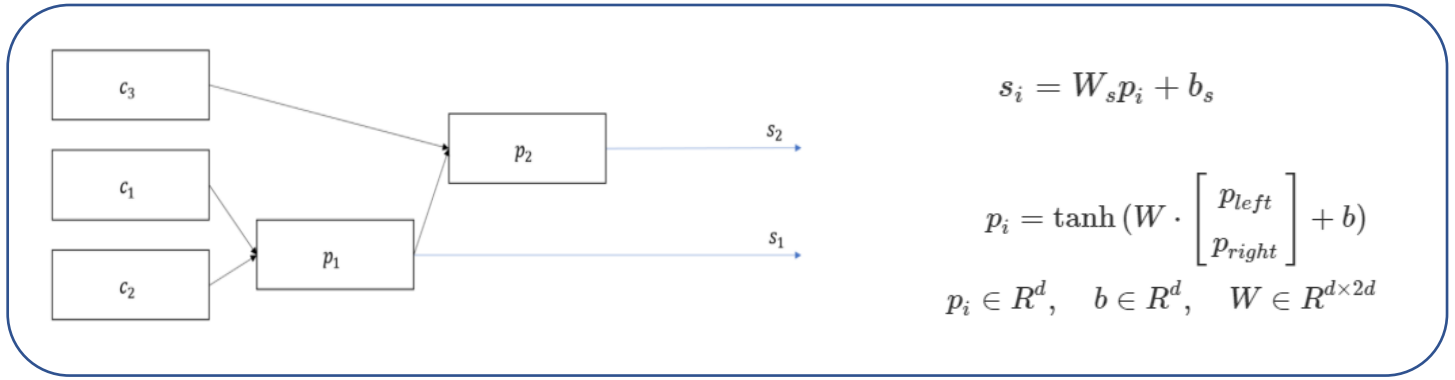
$$s_i = W_s p_i + b_s$$

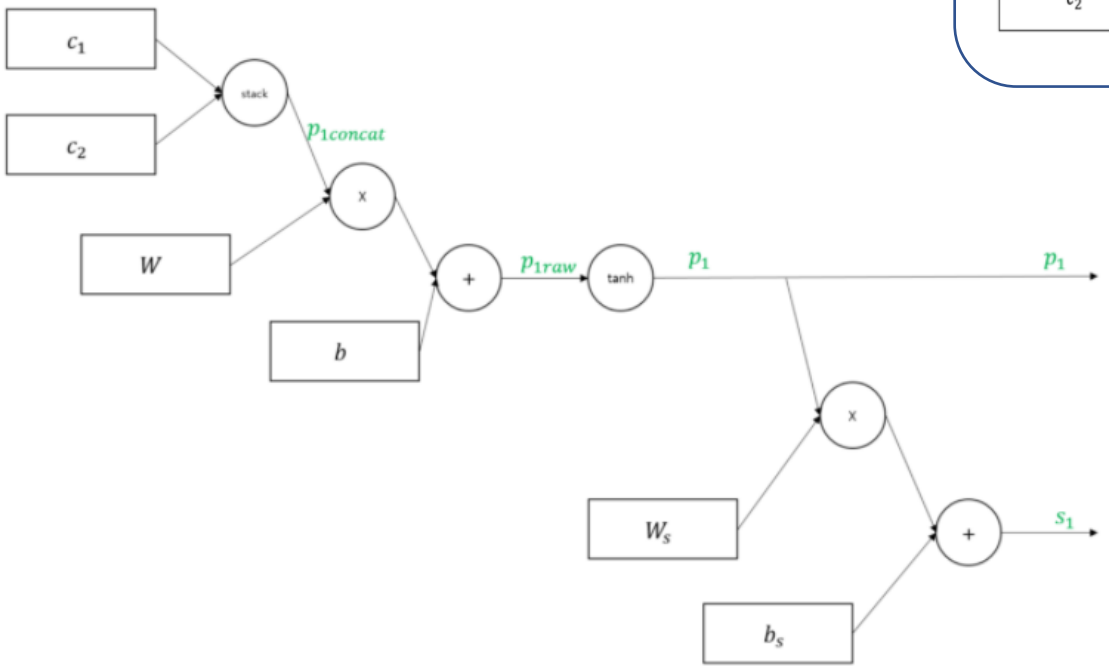
$$p_i = \tanh\left(W \cdot \begin{bmatrix} p_{left} \\ p_{right} \end{bmatrix} + b\right)$$

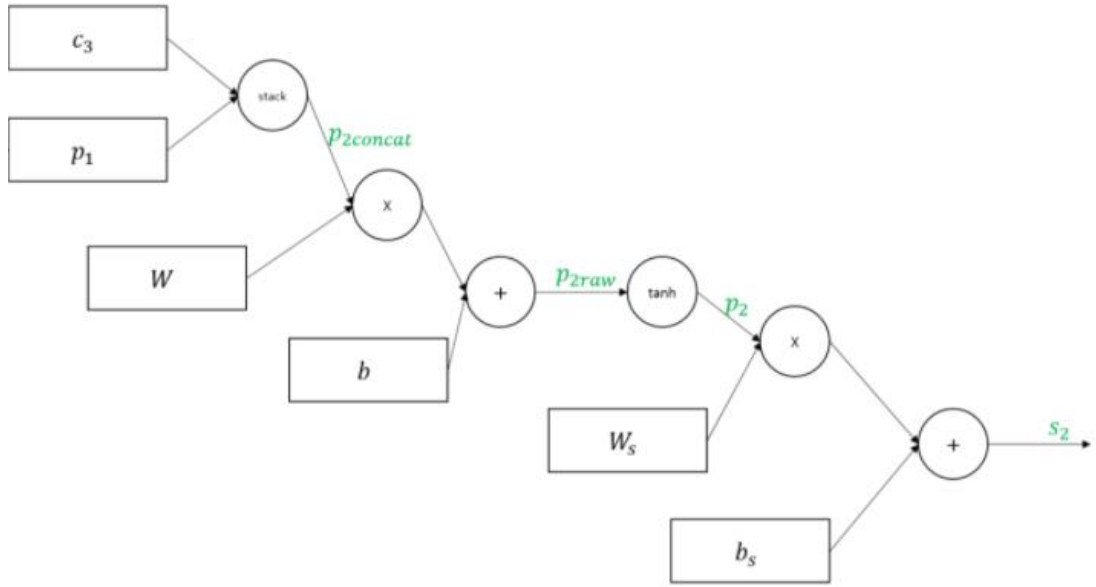
$$p_i \in R^d, \quad b \in R^d, \quad W \in R^{d \times 2d}$$

Simple Tree RNN

Forward pass



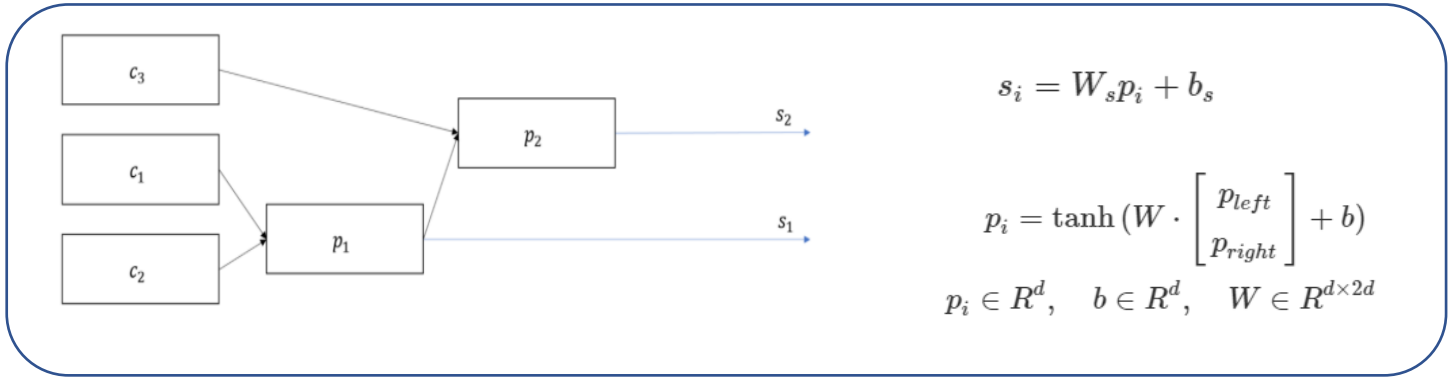


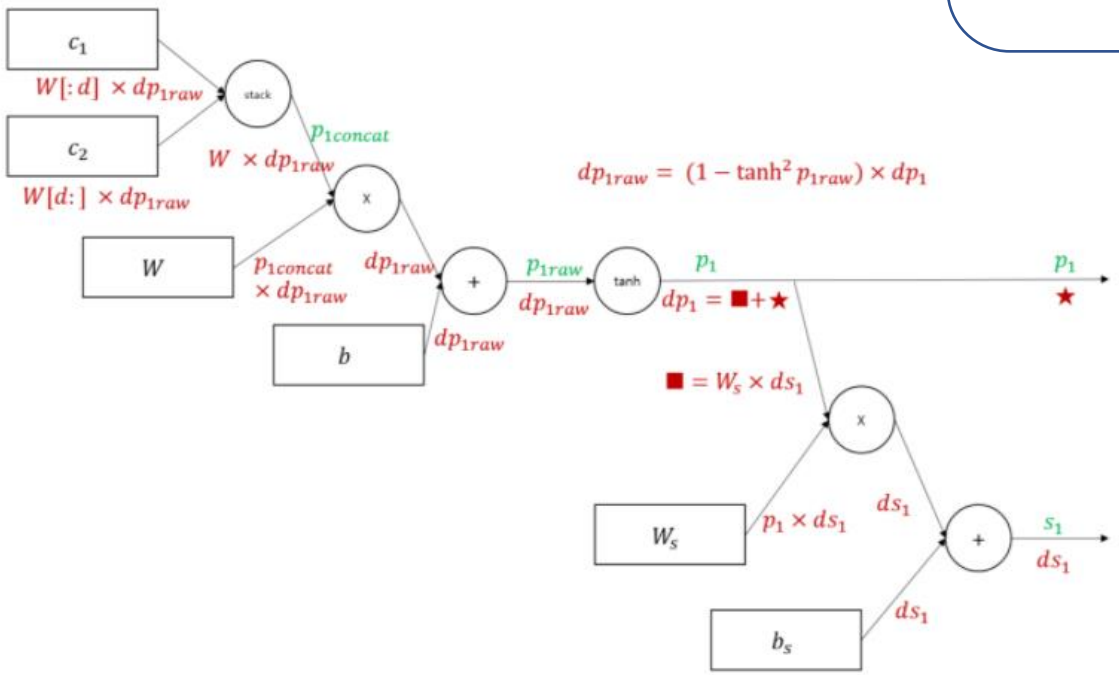


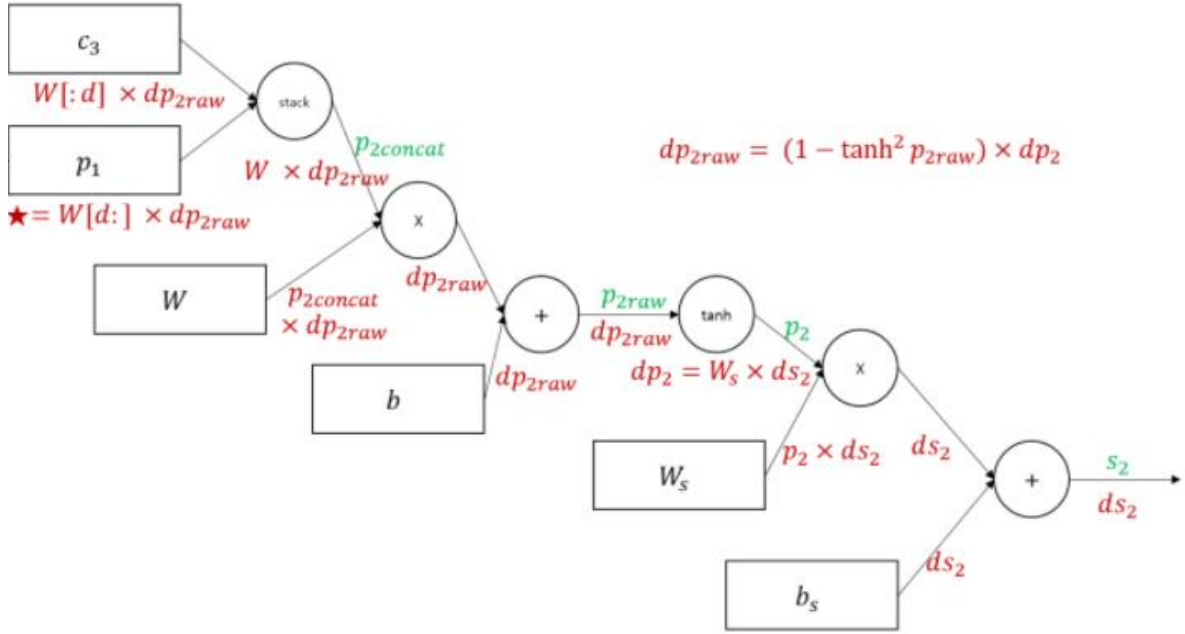
Simple Tree RNN

Backward pass

부모노드로부터 계산된 스코어와 해당 부분에 해당하는 정답 레이블과 비교한 후 오차를 최소화하는 방향으로 **역전파(backpropagation)**를 수행해 파라메터(W, b, Ws, bs)를 업데이트하는 방식으로 학습







Simple Tree RNN

Discussion

- Single weight matrix TreeRNN could capture some phenomena but not adequate for more complex, higher order composition and parsing long sentences.

(앞서 W가 모든 노드에서 동일하다고 했기 때문에 Simple TreeRNN은 일부 현상에서 적합할 수 있지만 더 복잡하고, 고차 구성 및 긴 문장에서는 적절치 못하다.)

- There is no real interaction between the input words.

(인풋 단어가 실제 상호작용을 모델링 한 것이 아니다. W는 인풋과 단순한 multiply이기 때문에 인풋 단어들 사이의 interaction을 modeling한 것이 아니다.)

- The composition function I the same for all syntactic categories, etc.

(조합 함수가 모든 경우에 대해서 동일하게 작용한다.)

Syntatically-Untied RNN

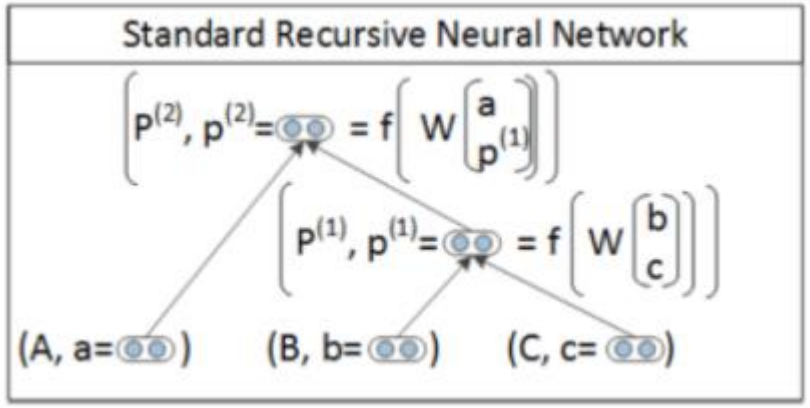
Definition

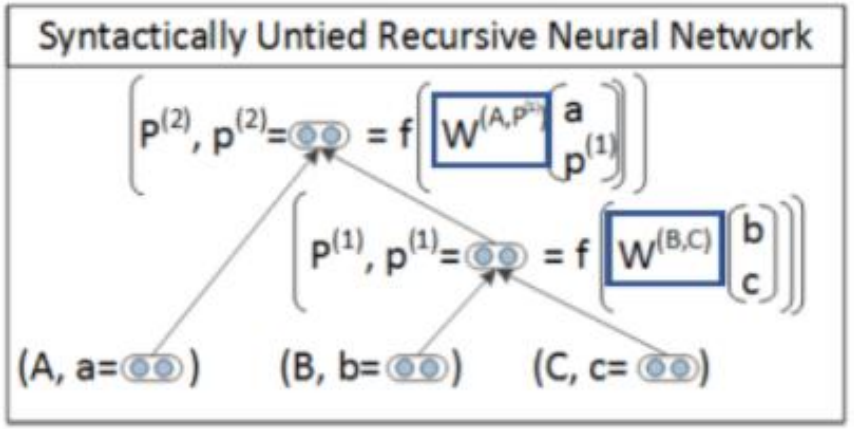
- **Idea**

Simple Tree RNN의 문제가 동일한 W를 쓰는 거였기 때문에
Children의 syntactic category를 이용하여 다른 composition matrix를 사용하자

- **Effect**

- Do better with different composition matrix for different syntactic environments
- Better semantics





Syntatically-Untied RNN

Probabilistic Context Free Grammar(PCFG)

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

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

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

Lexicon

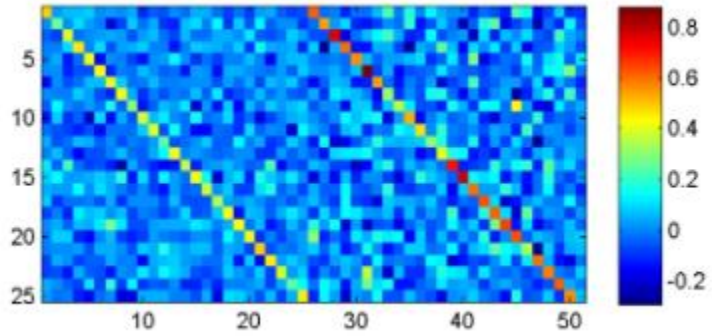
Det → the | a | that | this
0.6 0.2 0.1 0.1
Noun → book | flight | meal | money
0.1 0.5 0.2 0.2
Verb → book | include | prefer
0.5 0.2 0.3
Pronoun → I | he | she | me
0.5 0.1 0.1 0.3
Proper-Noun → Houston | NWA
0.8 0.2
Aux → does
1.0
Prep → from | to | on | near | through
0.25 0.25 0.1 0.2 0.2

이렇게 조합될 단어의 확률을 계산하고 TreeRNN에 적용한 모델이 **Syntactically-United RNN** 이다.

Syntatically-Untied RNN

Weight of SU-RNN

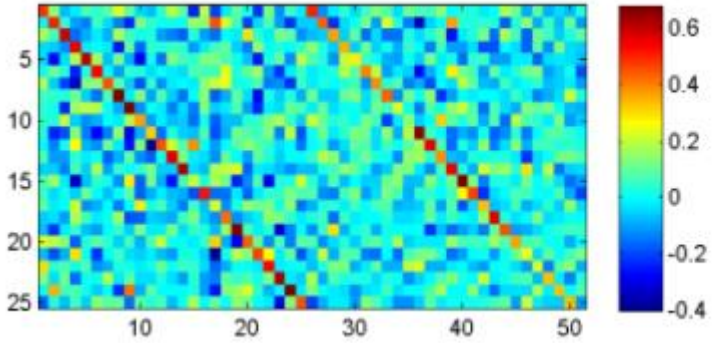
DT(관사)-NP(명사구)



DT-NP

관사보다는 명사구를 중요하게 취급
Ex) ‘a cat’, ‘the dog’

VP(동사구)-NP(명사구)



VP-NP

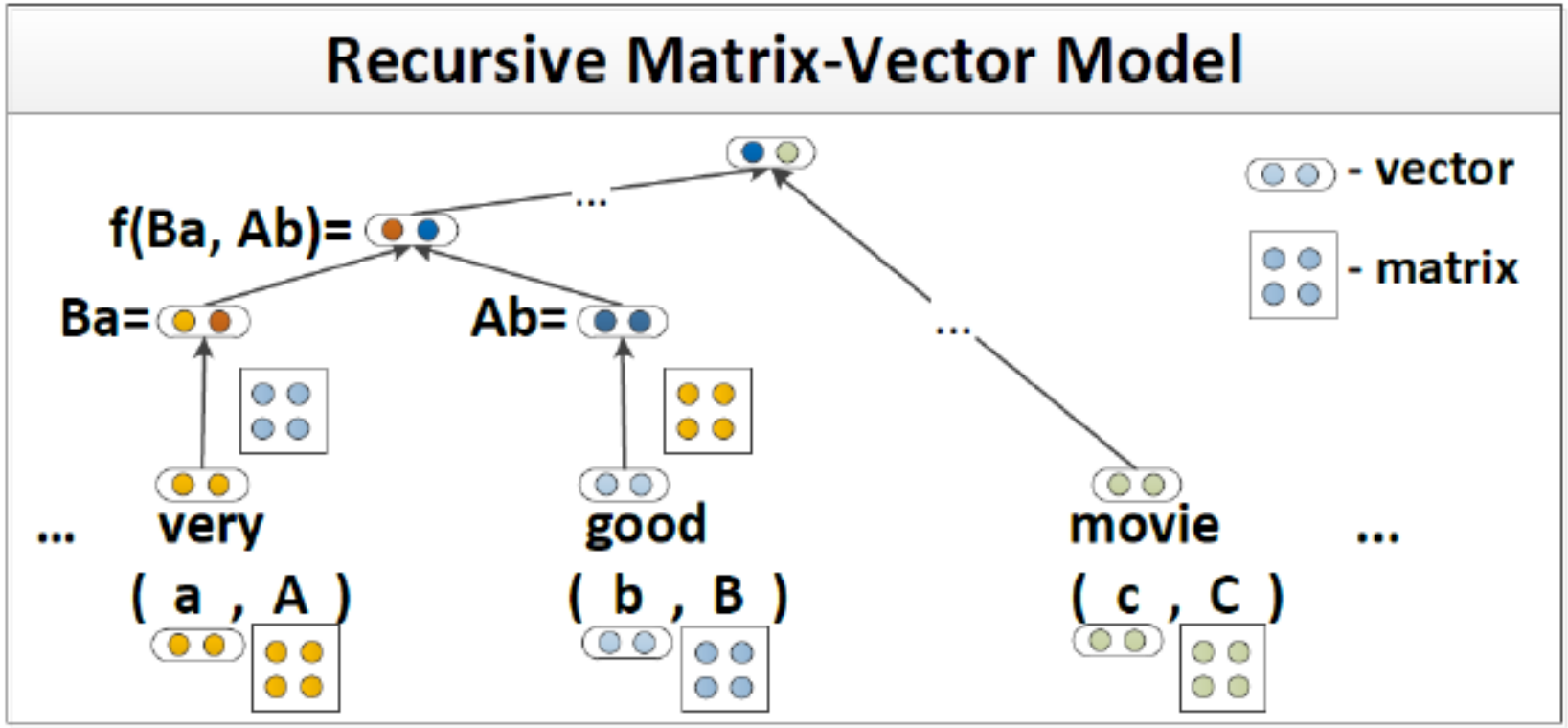
둘 모두 중요하게 취급
Ex) ‘eat food’, ‘push her’

초기에 a pair of didagonal matrix로 초기화를 한 후,
SU-RNN으로 학습한 가중치를 시각화한 그림이다. 붉은색일수록 그 가중치가 높다.

Compositionality Through Recursive Matrix-Vector Spaces

MV-RNN

결합하는 단어가 다르면 그 결합 과정 또한 다를 것이라는 전제가 깔려 있는 기법
따라서 단어벡터 뿐만 아니라 각 단어에 대한 행렬도 곱해준다.



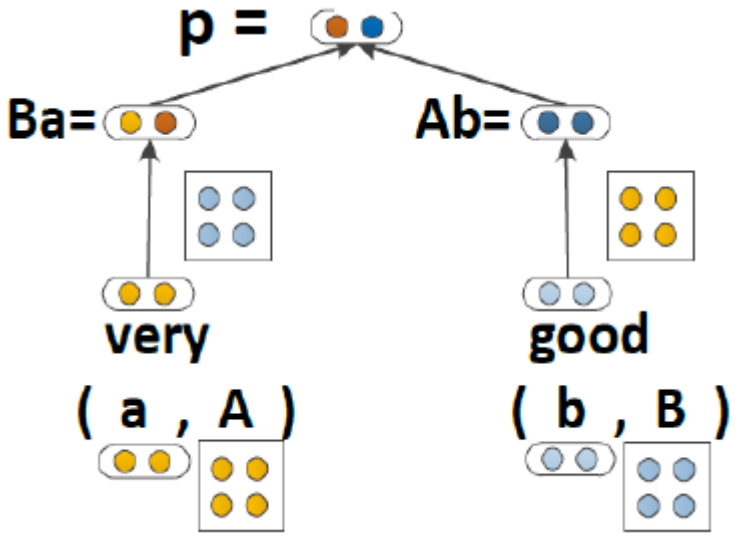
Compositionality Through Recursive Matrix-Vector Spaces

MV-RNN

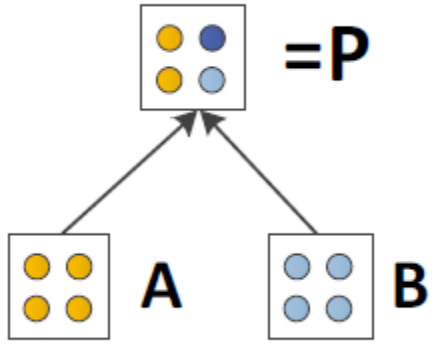
모든 단어, 모든 parent node에 대해 단어벡터+행렬을 설정

a, b, c, p ∈ R^d, A, B, C ∈ R^{d*d}, W ∈ R^{d*2d}, P_m ∈ R^{d*d}, W_m ∈ R^{d*2d}

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



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



Compositionality Through Recursive Matrix-Vector Spaces

Problems

- It introduced a humongous number of parameters.
(행렬도 사용하기 때문에 파라미터의 수가 매우 많아져 cost 비용이 높아진다.)

- We didn't have very good ways of sort of building up the matrix meaning of bigger phrases.
(큰 구절에 있어서 P_m을 만드는 방식이 좋은 방식은 아니다.)

Recursive Neural Tensor Network

Sentiment detection

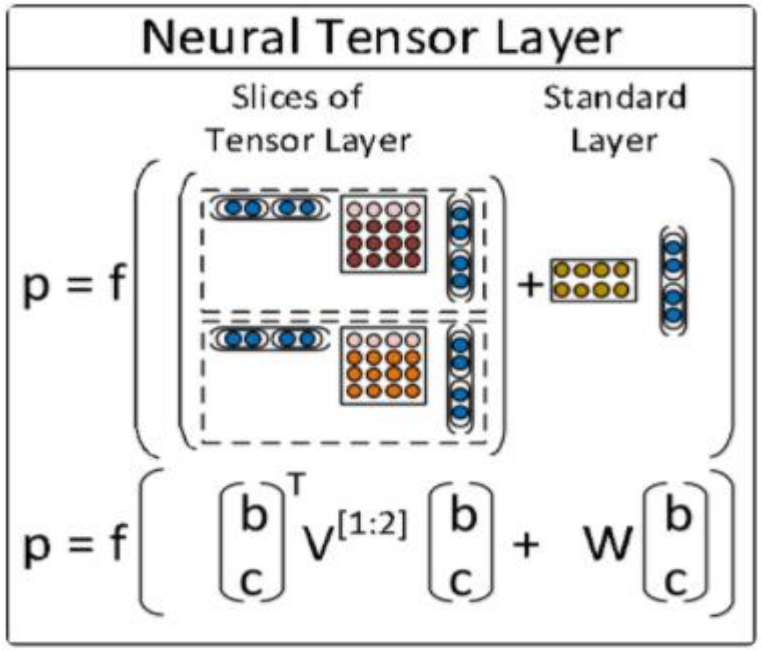
- 굳이 TreeRNN을 통해 단어, 문장을 분석하지 않고 Bag of Words를 통해서 임베딩하여 문장을 분석하여도 90%정도의 성능을 보인다.
- 하지만 다음과 같은 경우(negated positive)는 모델이 단어의 감정을 분석하기 어렵다.
- 이를 분석하기 위해 TreeRNN을 사용

With this cast, and this subject matter, the movie should have been funnier and more entertaining.

Recursive Neural Tensor Network

RNTN

- MV-RNN의 계산복잡성을 완화하기 위해 제안된 기법
- 모든 단어와 parent node에 대해 relational matrix를 생성하는 대신 모든 단어와 parent node가 공유하는 Tensor를 설정
- V가 Tensor를 의미(아래 그림에서는 2층짜리 행렬)



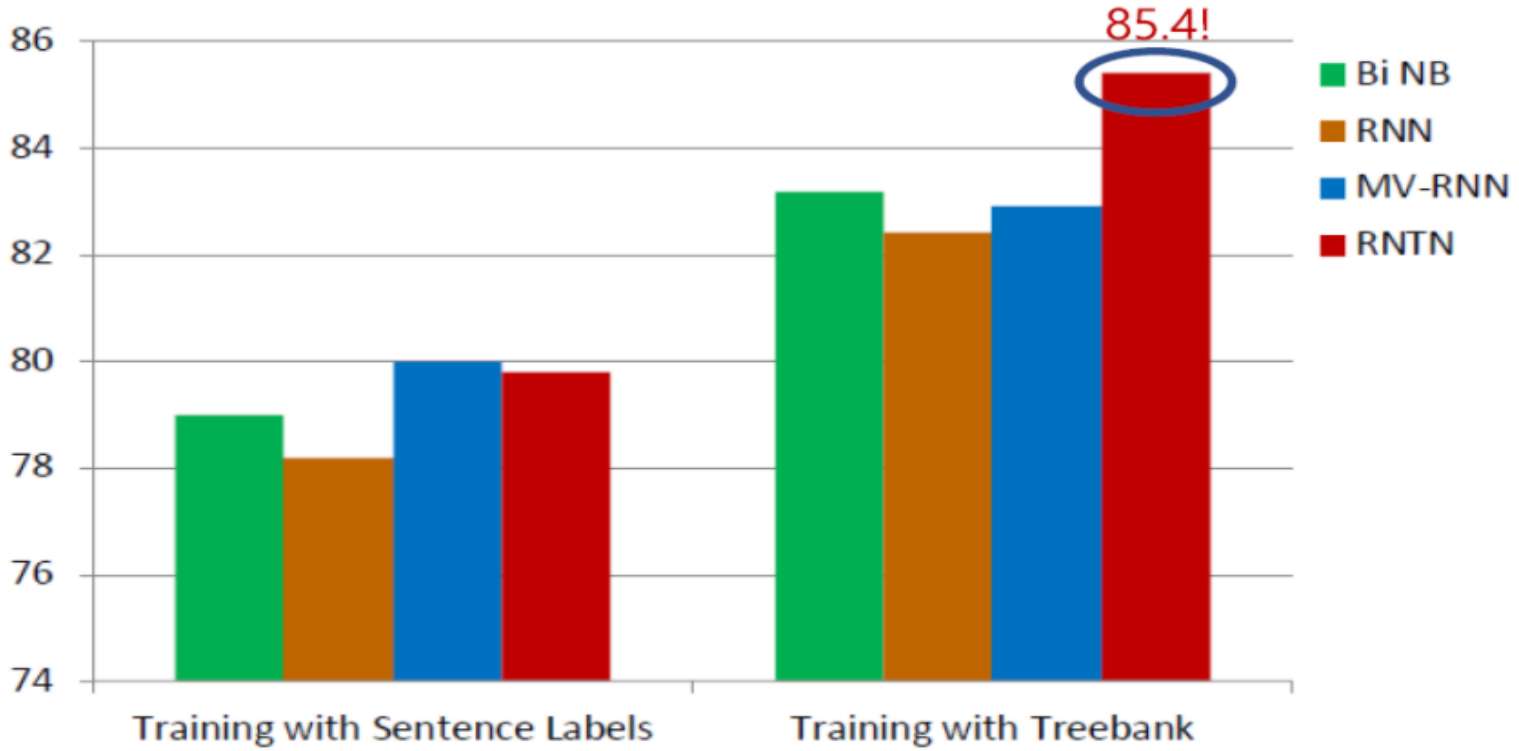
b, c ∈ R^{1*d}, [b, c] ∈ R^{1*2d}. V ∈ R^{2d*2d}

$$p = g\left(\begin{bmatrix} b \\ c \end{bmatrix}^T V^{[1:d]} \begin{bmatrix} b \\ c \end{bmatrix} + W \begin{bmatrix} b \\ c \end{bmatrix}\right)$$

- Slice of Tensor Layer 가운데 1층의 결과물은 스칼라값 하나가 나옴
- 이를 확장해 텐서가 d층이라고 가정하면 Slice of Tensor Layers를 계산한 결과는 R^d차원의 벡터가 됨
- =SimpleTreeRNN에서의 차원수와 같음

Recursive Neural Tensor Network

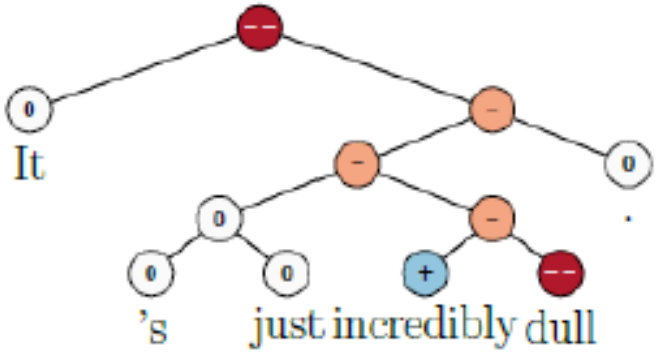
Positive/Negative results on Treebank

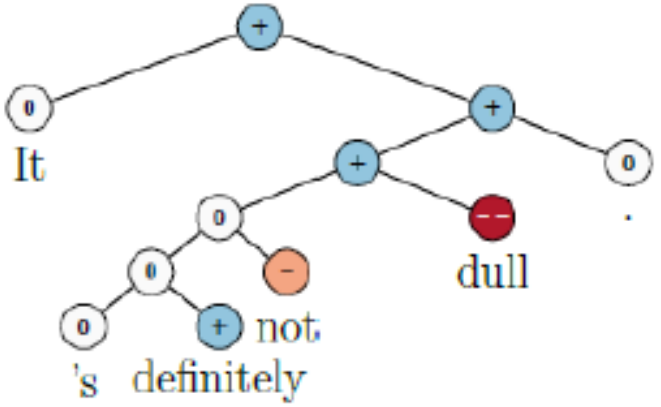


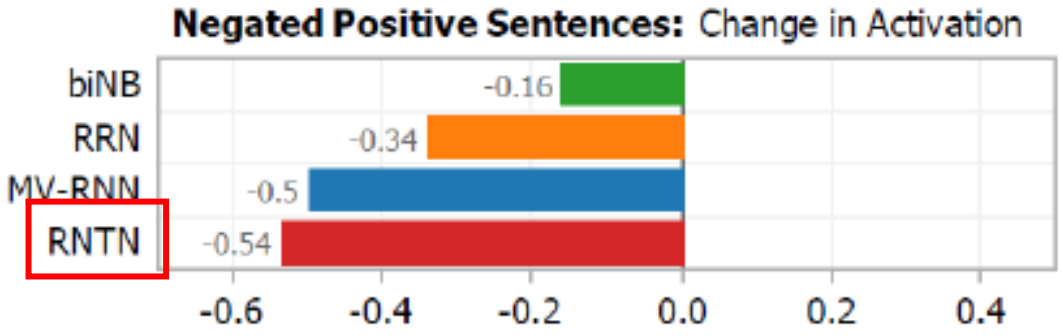
- Treebank 데이터는 구조적 분석에서 각 단어에 모두 라벨링이 되어있음
- RNTN이 다른 모델에 비해 높은 성능을 보임
- 제일 간단한 Bi NB 모델의 성능이 RNN, MV-RNN보다 높음=데이터셋의 중요성을 보여주는 지표

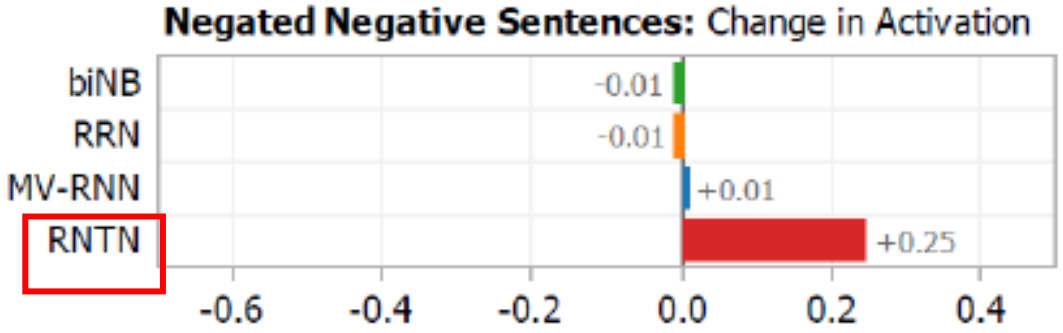
Recursive Neural Tensor Network

Negation Results









RNTN이 가장 Negation들을 잘 잡아내는 것을 볼 수 있다.

Recursive Neural Tensor Network

기법별 학습파라메터 비교

MV-RNN이 학습파라메터 종류가 많고 크기가 제일 큼
simple RNN은 가장 적고,
RNTN은 둘의 중간 정도

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	$W : d \times 2d$	$W : d \times 2d,$ $W_M : d \times 2d,$ $A, B, C \ldots : d \times d \times \lvert v \rvert$	$W : d \times 2d,$ $V^{[1:d]} : 2d \times 2d \times d$

Recursive Neural Tensor Network

Problems in Application

현실적으로 TreeRNN은 사용하기 힘들다.

- GPU 연산이 힘들다.

-병렬적으로 연산이 진행되어야 하는데 모든 task 마다 연산 구조가 동일하지 않고

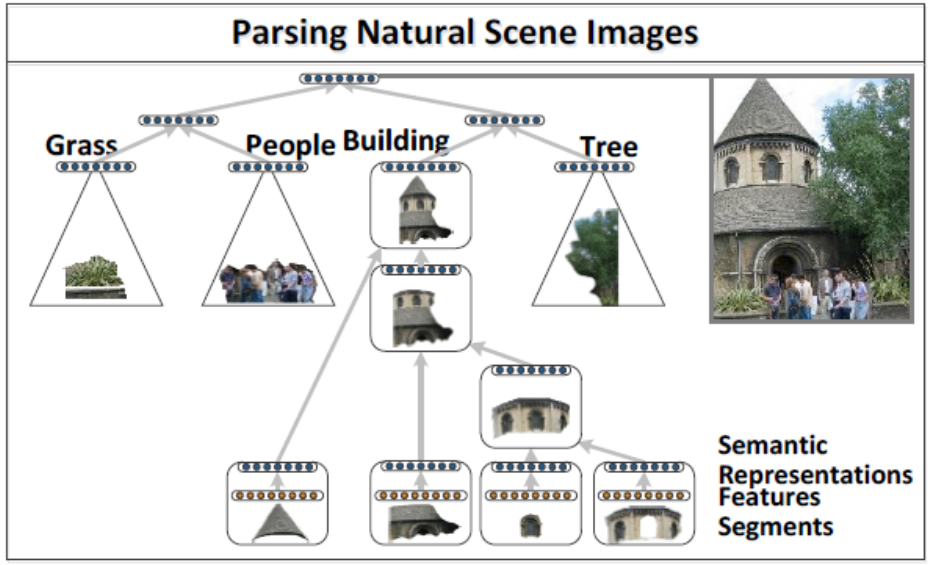
-Tree 모양이 다름

- 데이터에 라벨링을 모두 거쳐야 하는데 이 데이터를 구축하는데 어려움이 있다.

Application

Scene Parsing

Same Recursive Neural Network as for natural language parsing!
(Socher et al. ICML 2011)



Multi class segmentation

Method	Accuracy
Pixel CRF (Gould et al., ICCV 2009)	74.3
Classifier on superpixel features	75.9
Region-based energy (Gould et al., ICCV 2009)	76.4
Local labelling (Tighe & Lazebnik, ECCV 2010)	76.9
Superpixel MRF (Tighe & Lazebnik, ECCV 2010)	77.5
Simultaneous MRF (Tighe & Lazebnik, ECCV 2010)	77.5
Recursive Neural Network	**78.1**

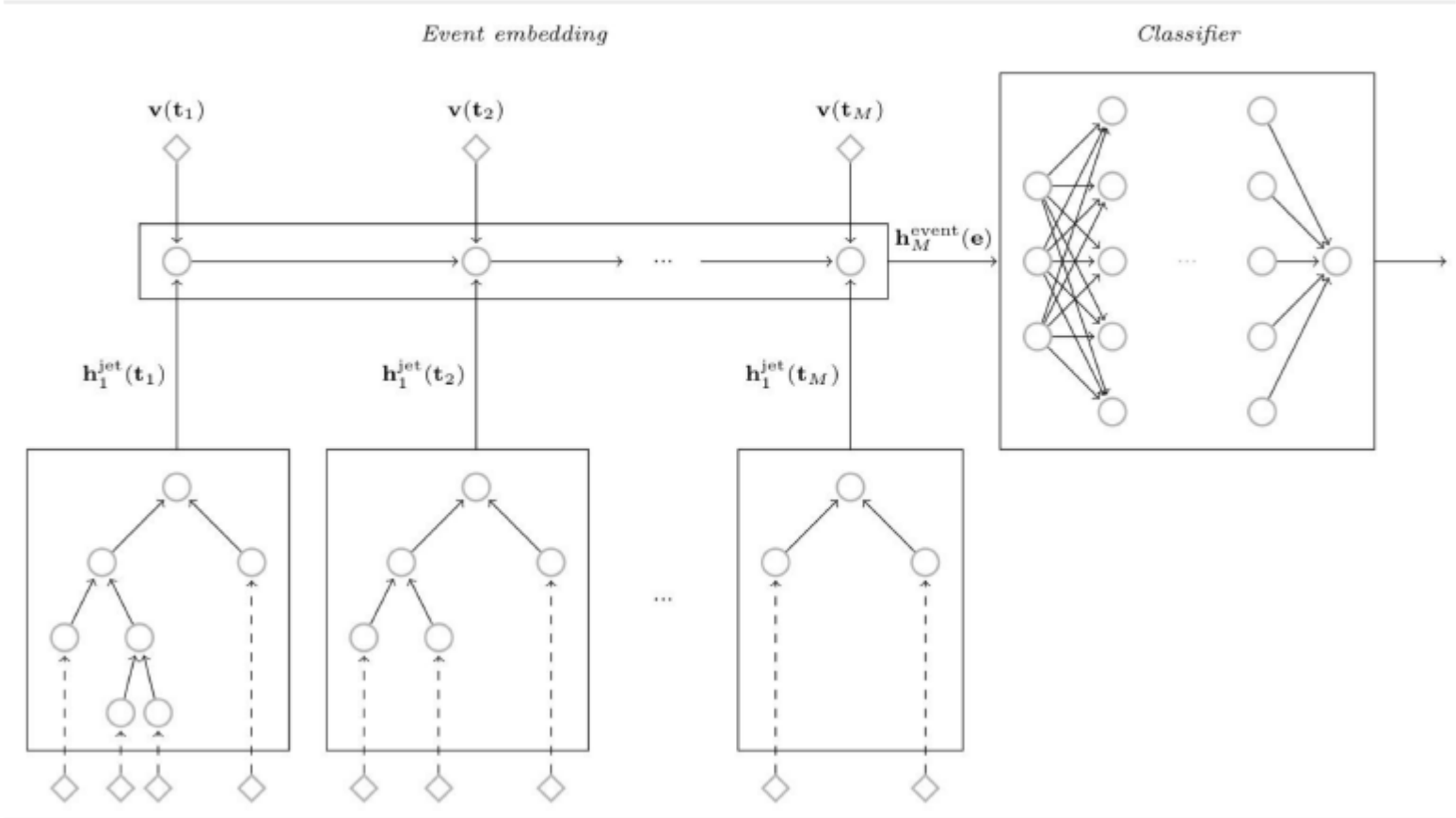
Application

QCD

물리에도 적용됨

5. QCD-Aware Recursive Neural Networks for Jet Physics

Gilles Louppe, Kyunghun Cho, Cyril Becot, Kyle Cranmer (2017)



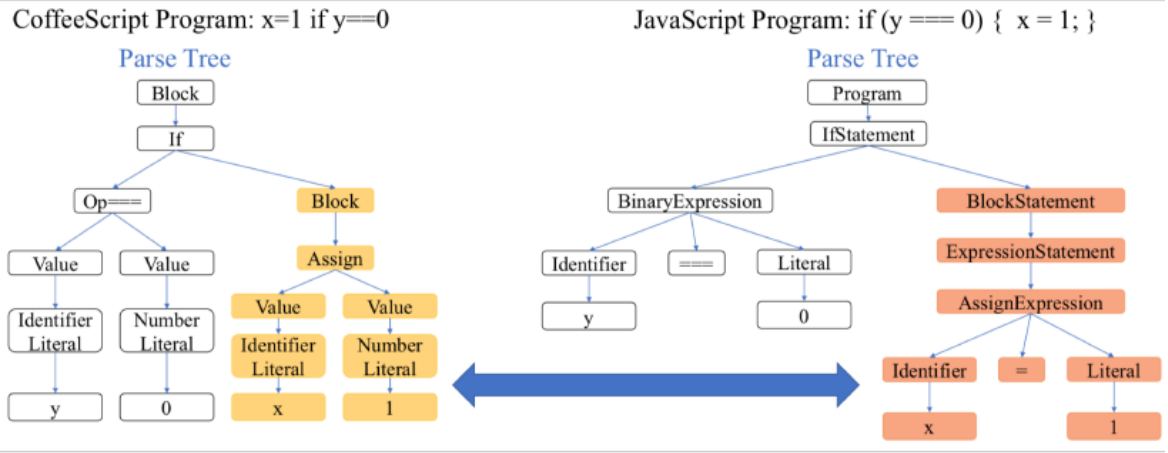
Application

Treee to tree NN for Program Trnaslation

Tree-to-tree Neural Networks for Program Translation

[Chen, Liu, and Song NeurIPS 2018]

- Explores using tree-structured encoding **and generation** for translation between programming languages
- In generation, you use attention over the source tree



	Tree2tree			Seq2seq				Seq2tree		Tree2seq	
	T→T	T→T (-PF)	T→T (-Attn)	P→P	P→T	T→P	T→T	P→T	T→T	T→P	T→T
CoffeeScript to JavaScript translation											
CJ-AS	**99.57%**	98.80%	0.09%	90.51%	79.82%	92.73%	89.13%	86.52%	88.50%	96.96%	92.18%
CJ-BS	**99.75%**	99.67%	0%	97.44%	16.26%	98.05%	93.89%	91.97%	88.22%	96.83%	78.77%
CJ-AL	**97.15%**	71.52%	0%	21.04%	0%	0%	0%	80.82%	78.60%	82.55%	46.94%
CJ-BL	**95.60%**	78.61%	0%	19.26%	9.98%	25.35%	42.08%	76.12%	76.21%	83.61%	26.83%
JavaScript to CoffeeScript translation											
JC-AS	**87.75%**	85.11%	0.09%	83.07%	86.13%	73.88%	86.31%	86.86%	86.99%	71.61%	86.53%
JC-BS	**86.37%**	80.35%	0%	80.49%	85.94%	69.77%	85.28%	85.06%	84.25%	66.82%	85.31%
JC-AL	**78.59%**	54.93%	0%	77.10%	77.30%	65.52%	75.70%	77.11%	77.59%	60.75%	75.75%
JC-BL	**75.62%**	44.40%	0%	73.14%	73.96%	61.92%	74.51%	74.34%	71.56%	57.09%	73.86%

Attention을 함께 적용했을 때 성능이 좋으며,
Attention을 제거했을 경우 성능이 0이 되버림.
Recursive NN의 가능성을 볼 수 있는 적용사례

Reference

https://velog.io/@tobigs-text1314/CS224n-Lecture-18-Constituency-Parsing-TreeRNNS

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

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

CS224N lecture 18: Constituency Parsing, Treee RNNS