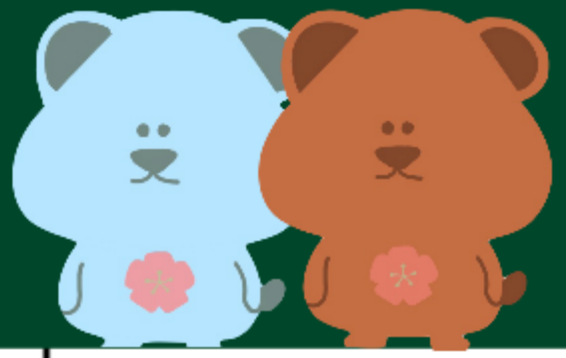
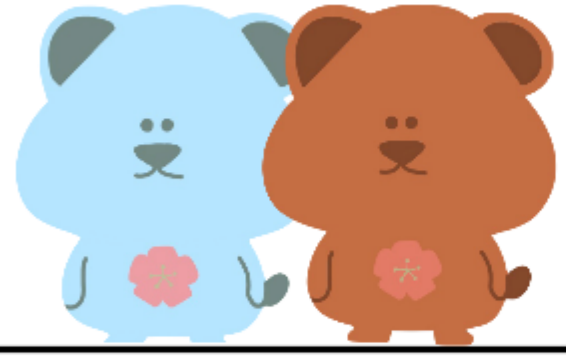


Week 18. Constituency Parsing and Tree Recursive Neural Networks

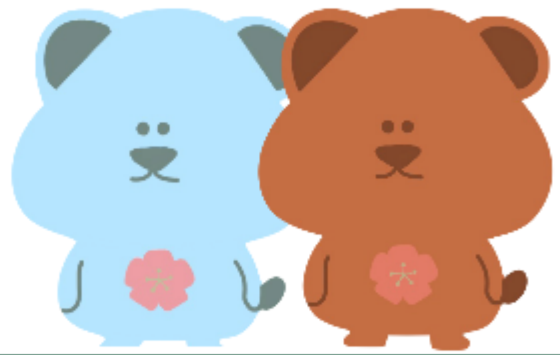
발표자: 김소민, 조서영

목차



EWHA EURON

Motivation: Compositionality and Recursion



Spectrum of Language in CS

NLP Models of Language
→ bag-of-words model : proven efficient

Linguistics → Emphasis on structure
→ exists a huge gap
→ good points exist in the middle (certain amounts of structure)

Working out the meaning of larger phrases?

The **snowboarder** is leaping over a mogul

vs

A **person on a snowboard** jumps into the air

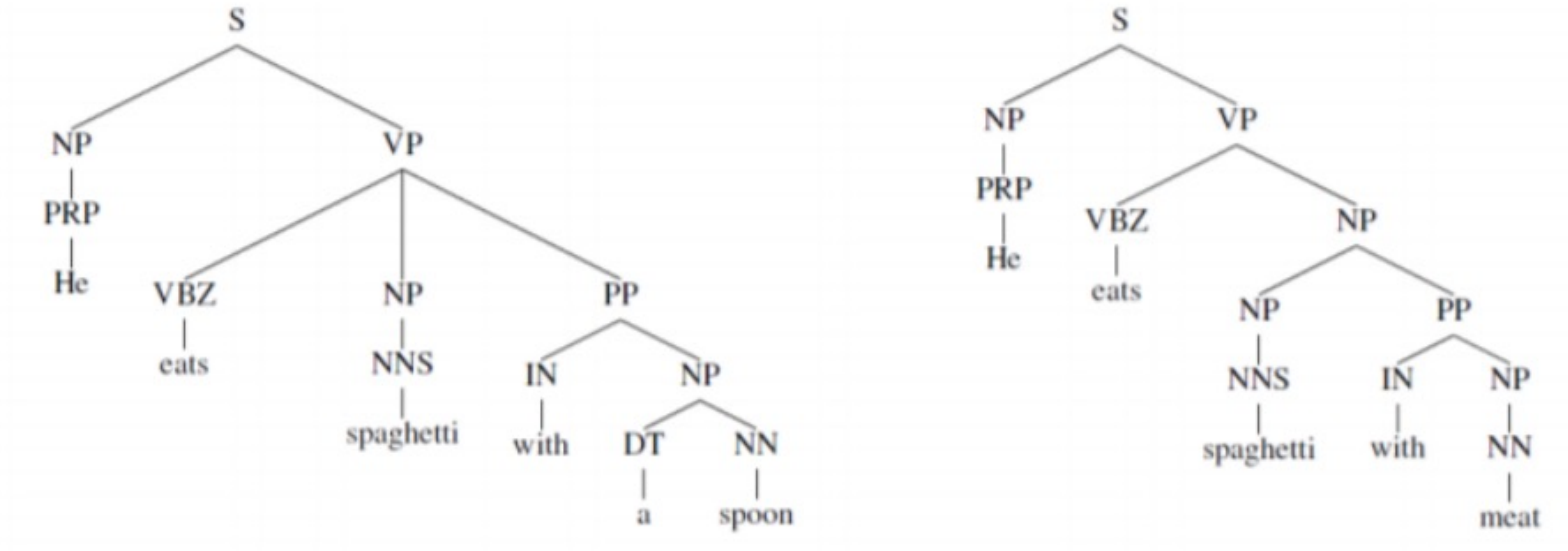
→ **Principle of Compositionality**

knowing meanings of components and putting the meaning together

Semantic composition of smaller elements

Working out the meaning of larger phrases?

→ Want a neural model that could use the hierarchical trees



Structure prediction with simple Tree RNN: Parsing

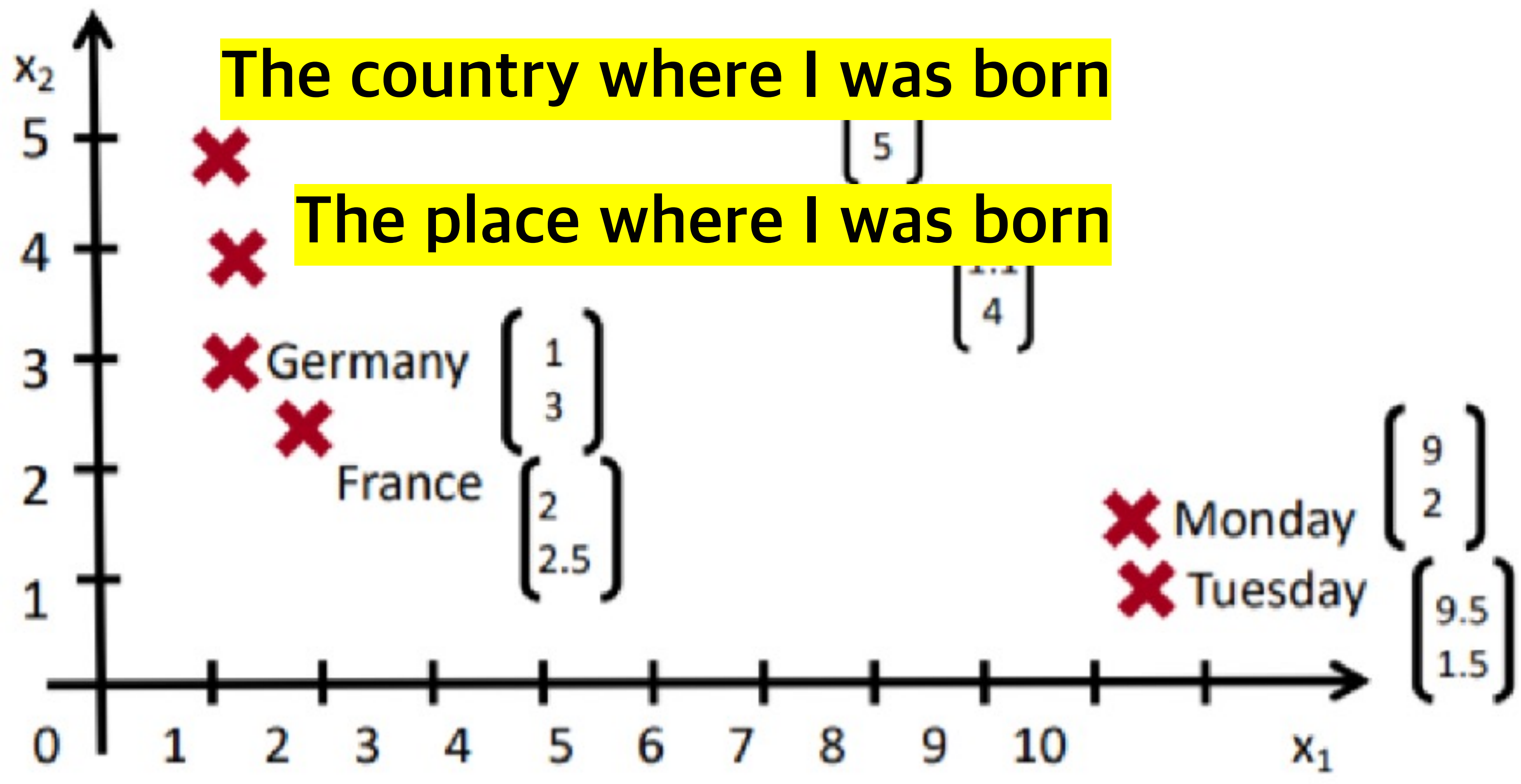
Determining Embeddings of phrases

Principle of Compositionality

→ The meaning (vector) of a sentence is determined by
1. the meaning of the words
2. the rules that combine them

objective is to put phrases into the same vector space as word embeddings

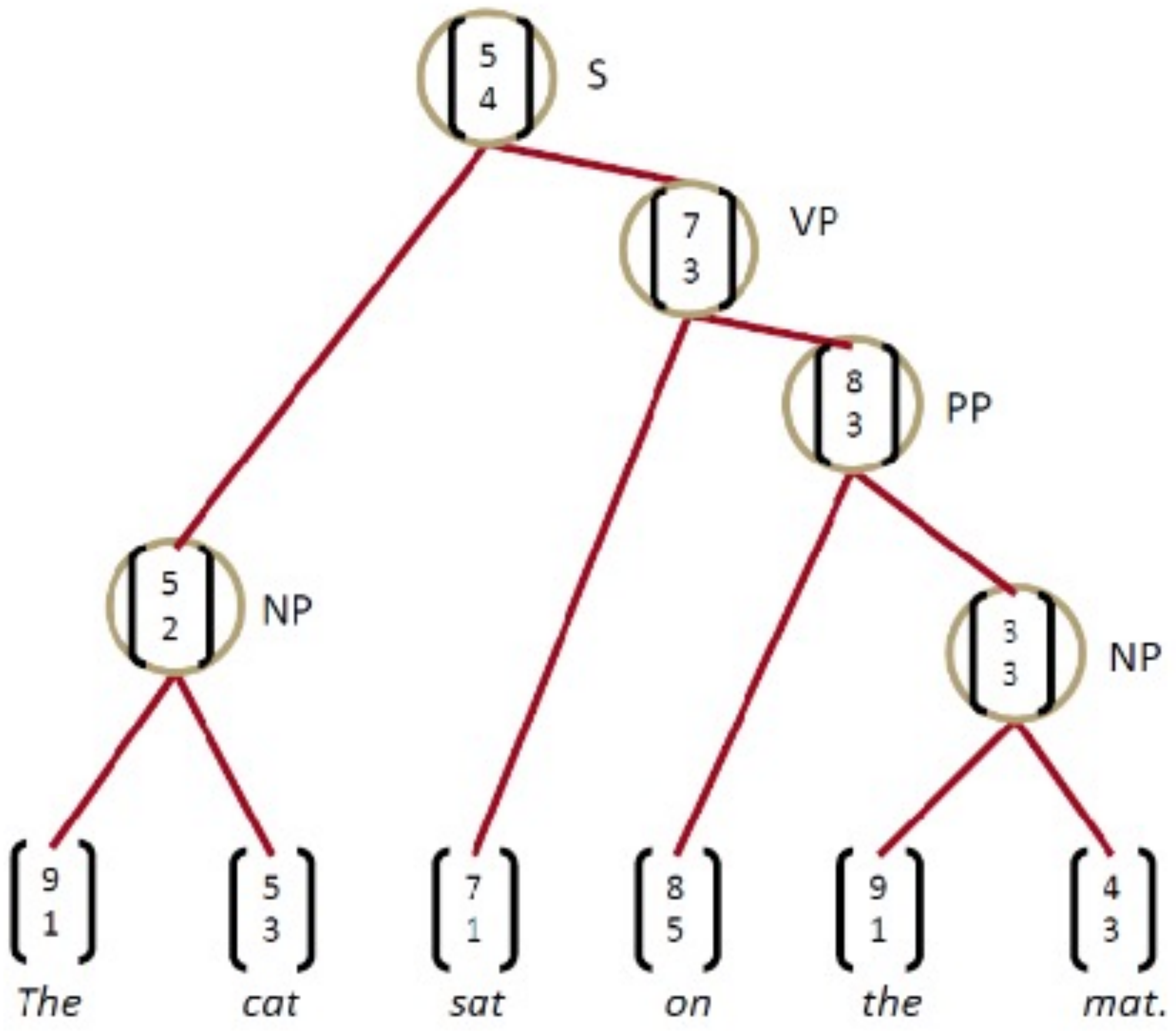
Determining Embeddings of phrases



objective is to put phrases into the same vector space as word embeddings

Determining Embeddings of phrases

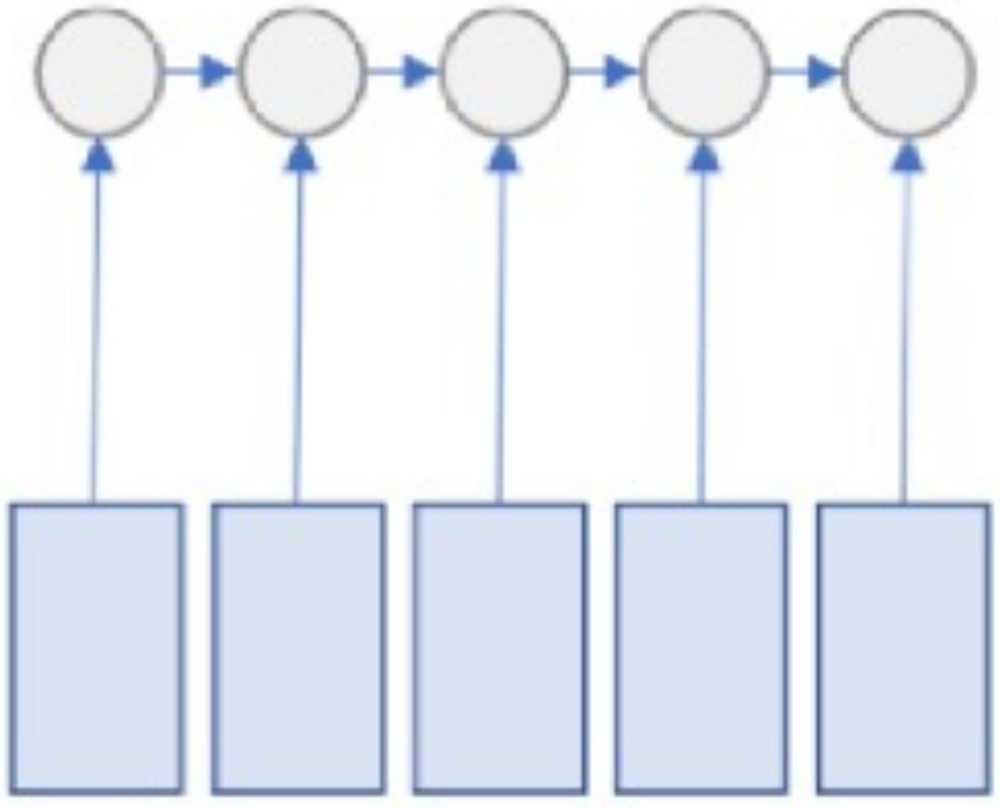
Have certain rules of
Combining components & create
vectors that contain meaning



Recursive vs RNN

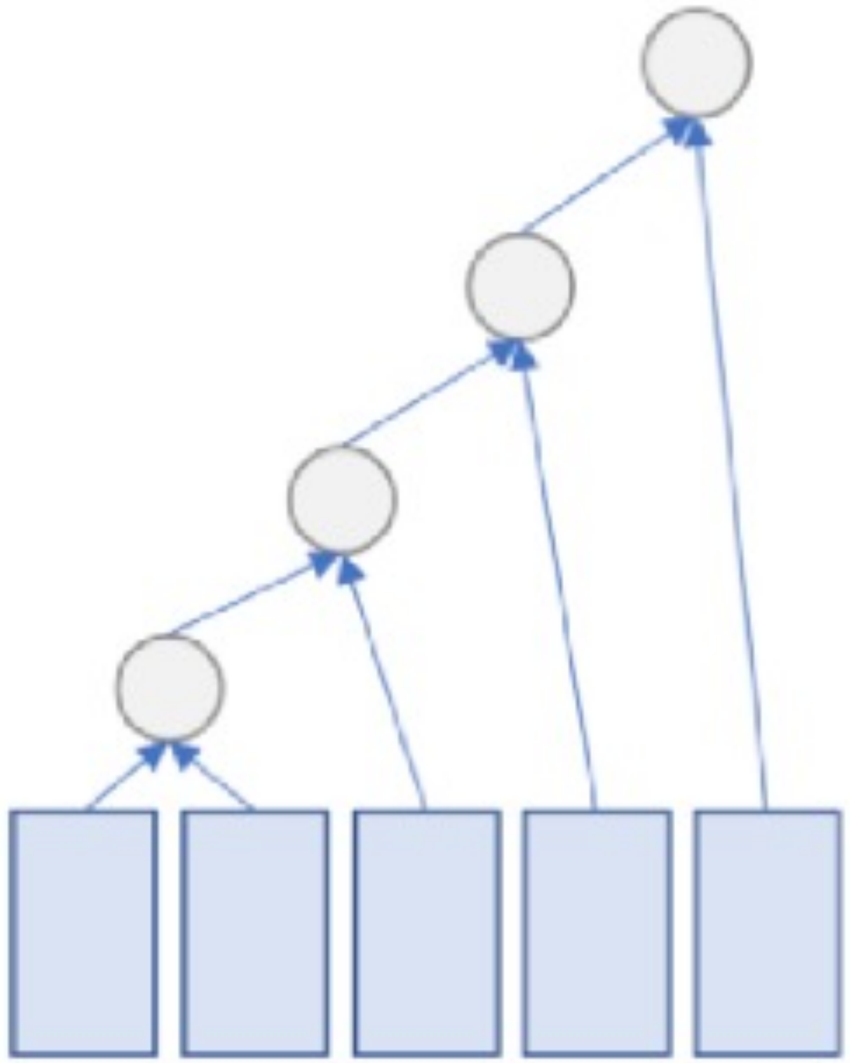
RNN

- can't capture phrases without prefix context
- often capture too much of last words in final vector



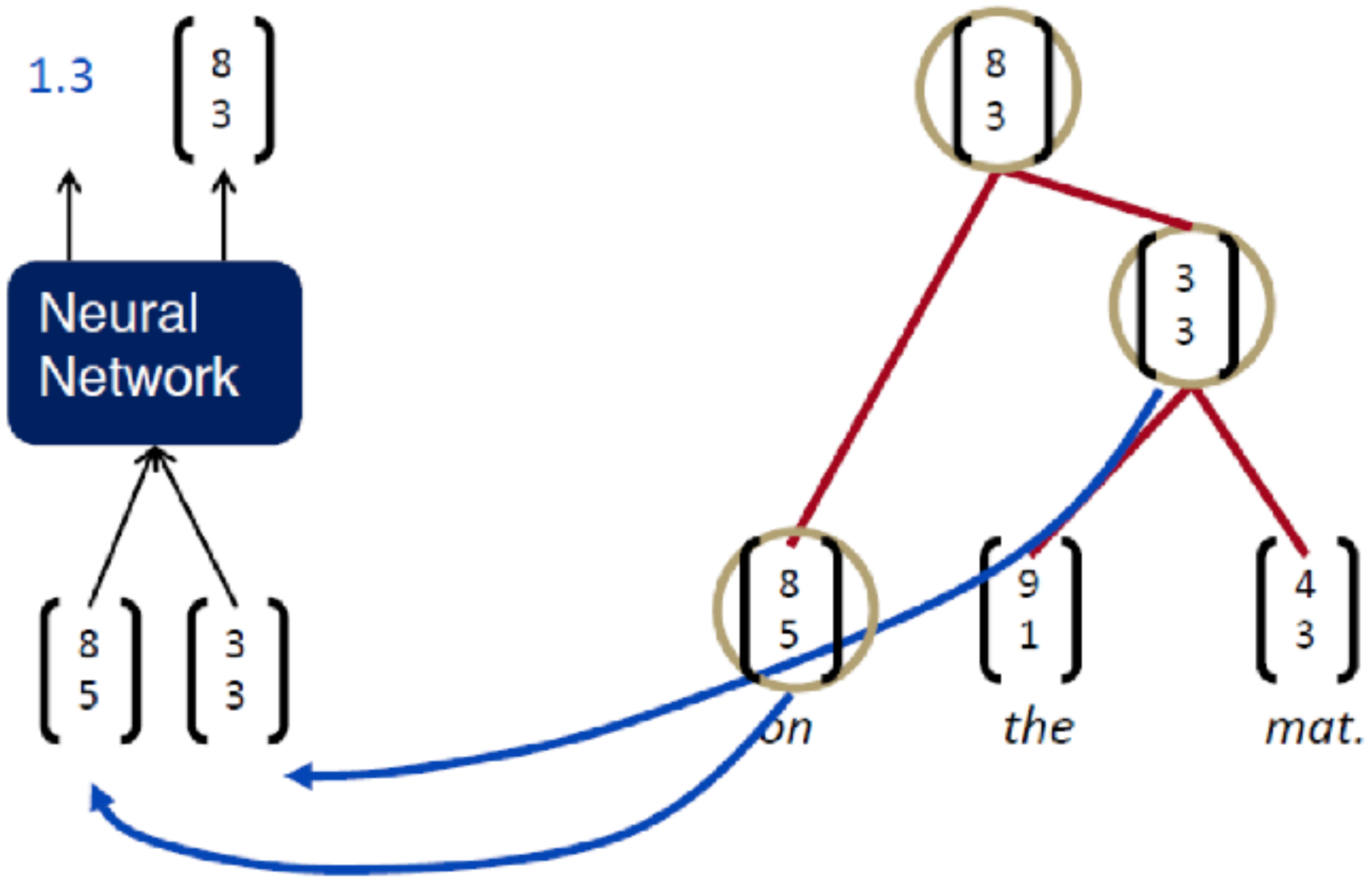
Recursive vs RNN

Recursive
- requires a tree structure (to know components)
- sensitive to its syntactic structure



Recursive

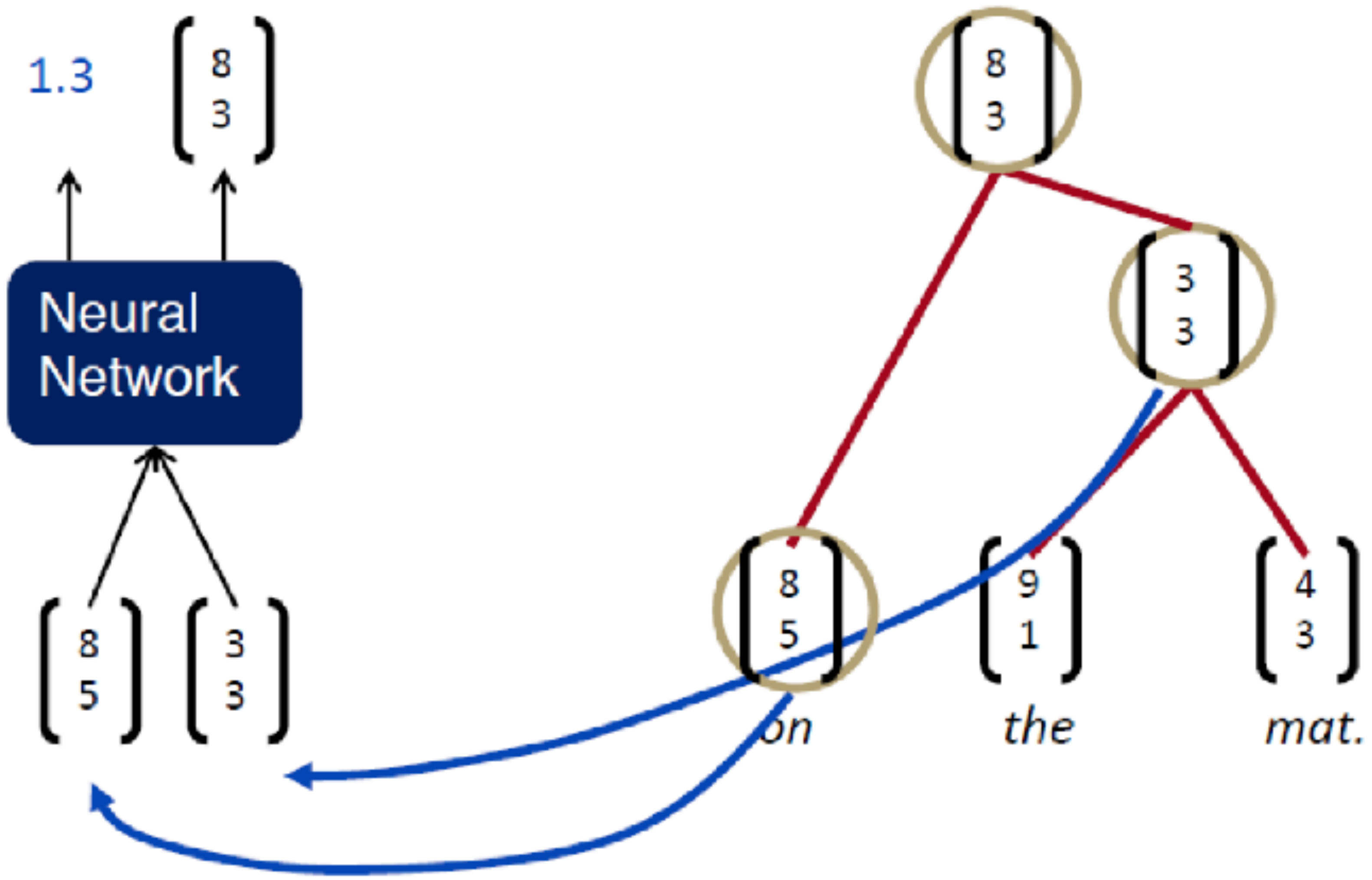
RNN for Structure Prediction



Input: two candidate children's representations

Output:
- Semantic representation if the two nodes are merged
- Score of how plausible the new node would be

RNN for Structure Prediction

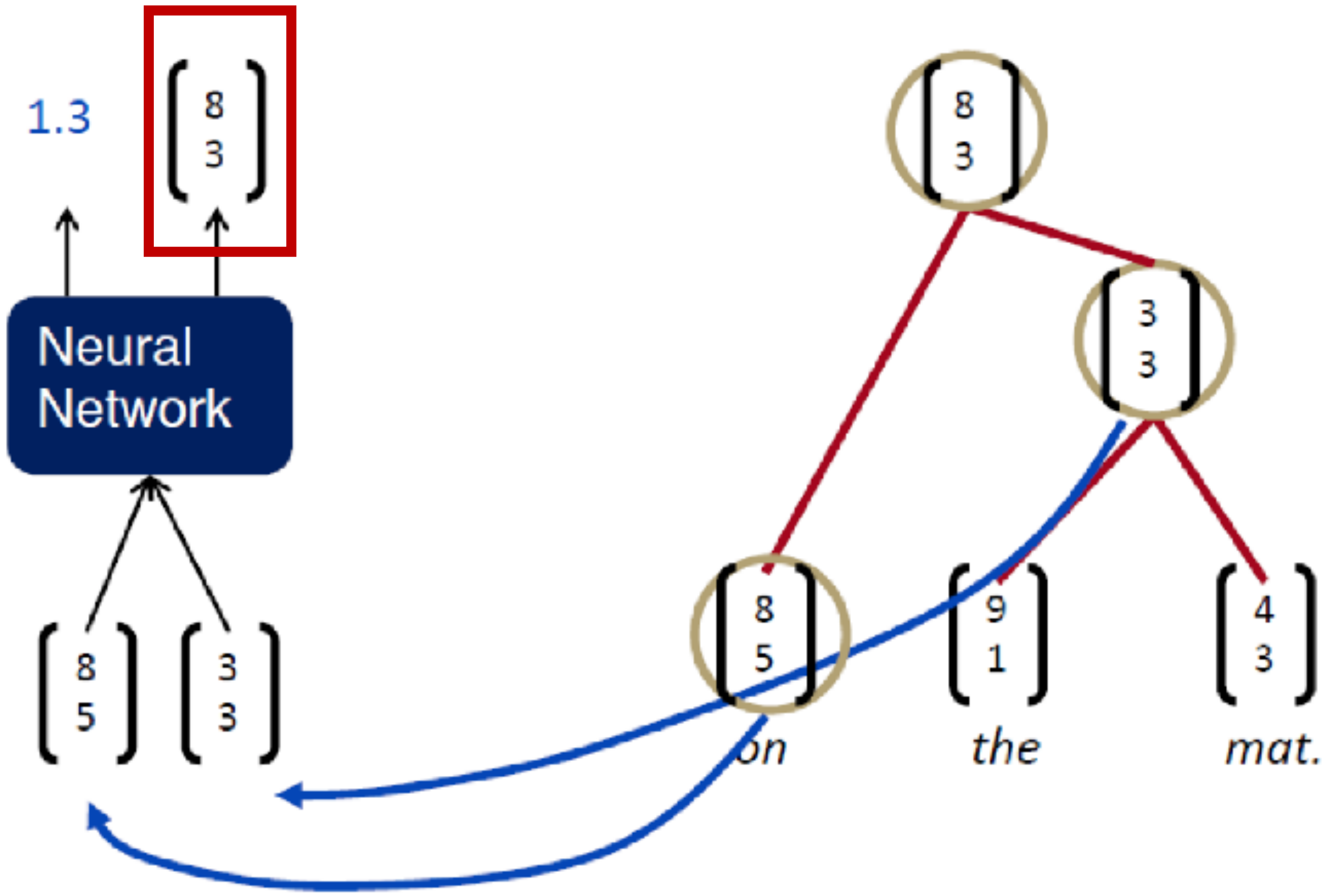


Input: two candidate children's representations

Output:

- Semantic representation if the two nodes are merged
- Score of how plausible the new node would be

RNN for Structure Prediction



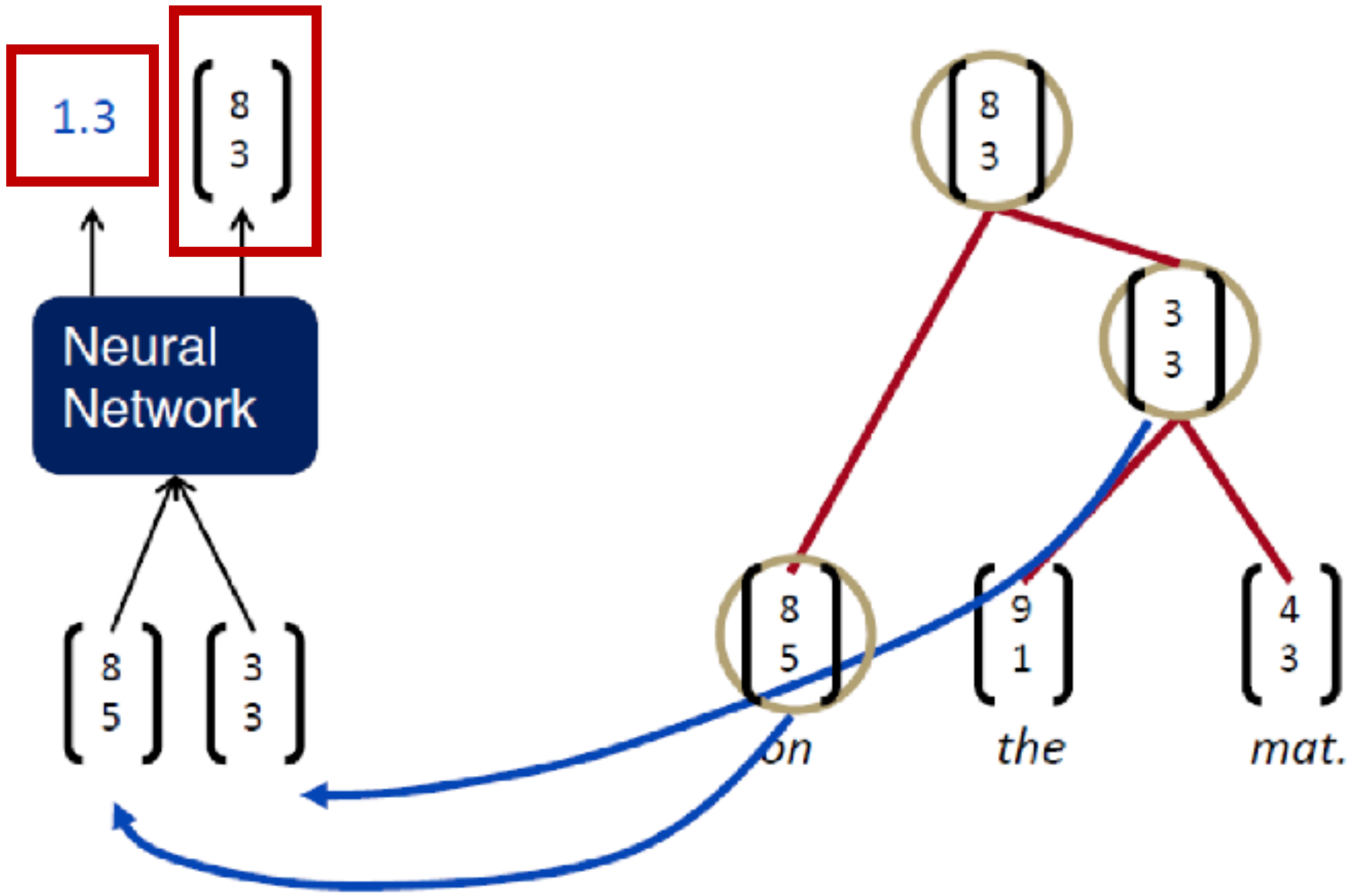
Output:

- Semantic representation if the two nodes are merged

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

- W same for all nodes

RNN for Structure Prediction



Output:

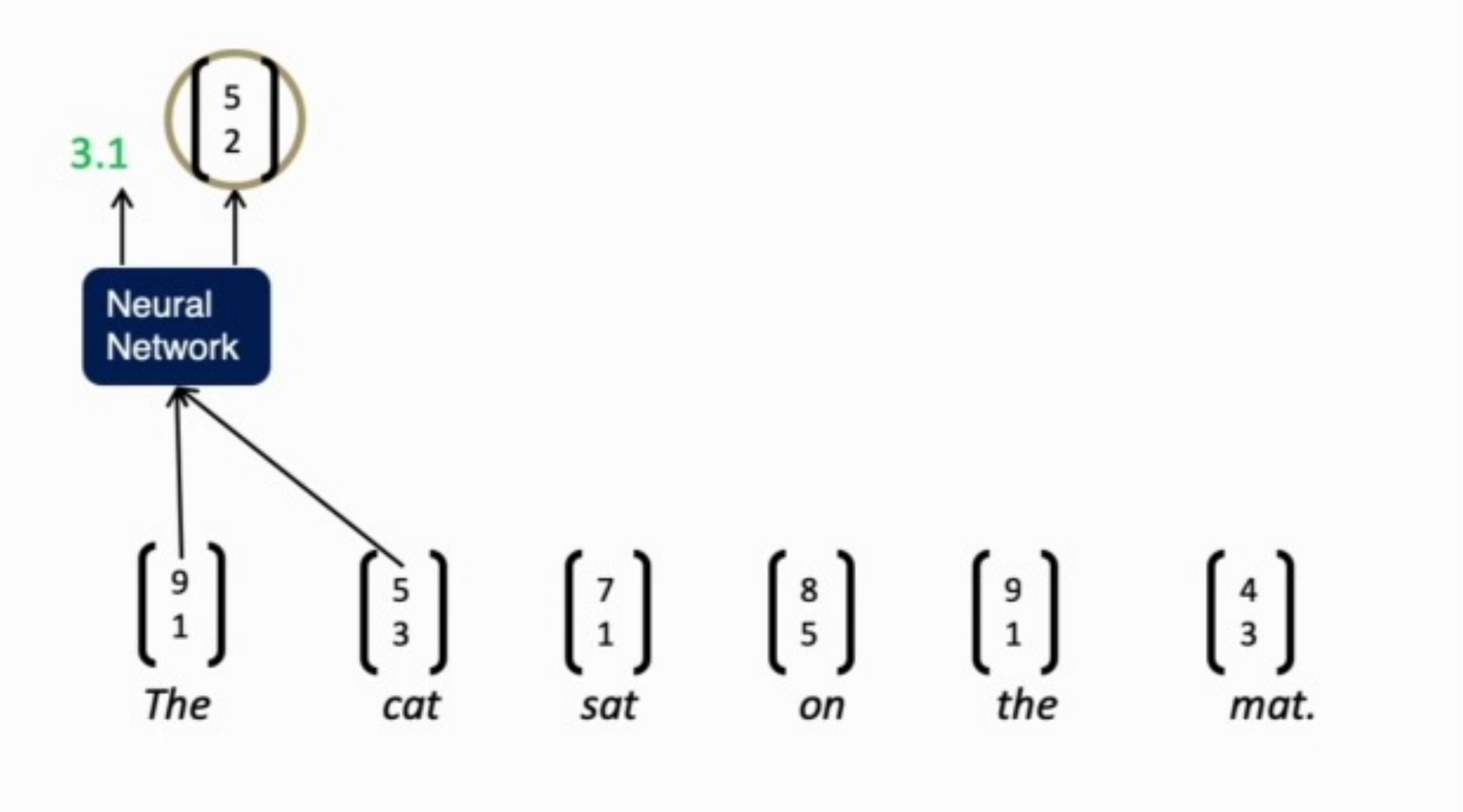
- Score of how plausible the new node would be

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

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

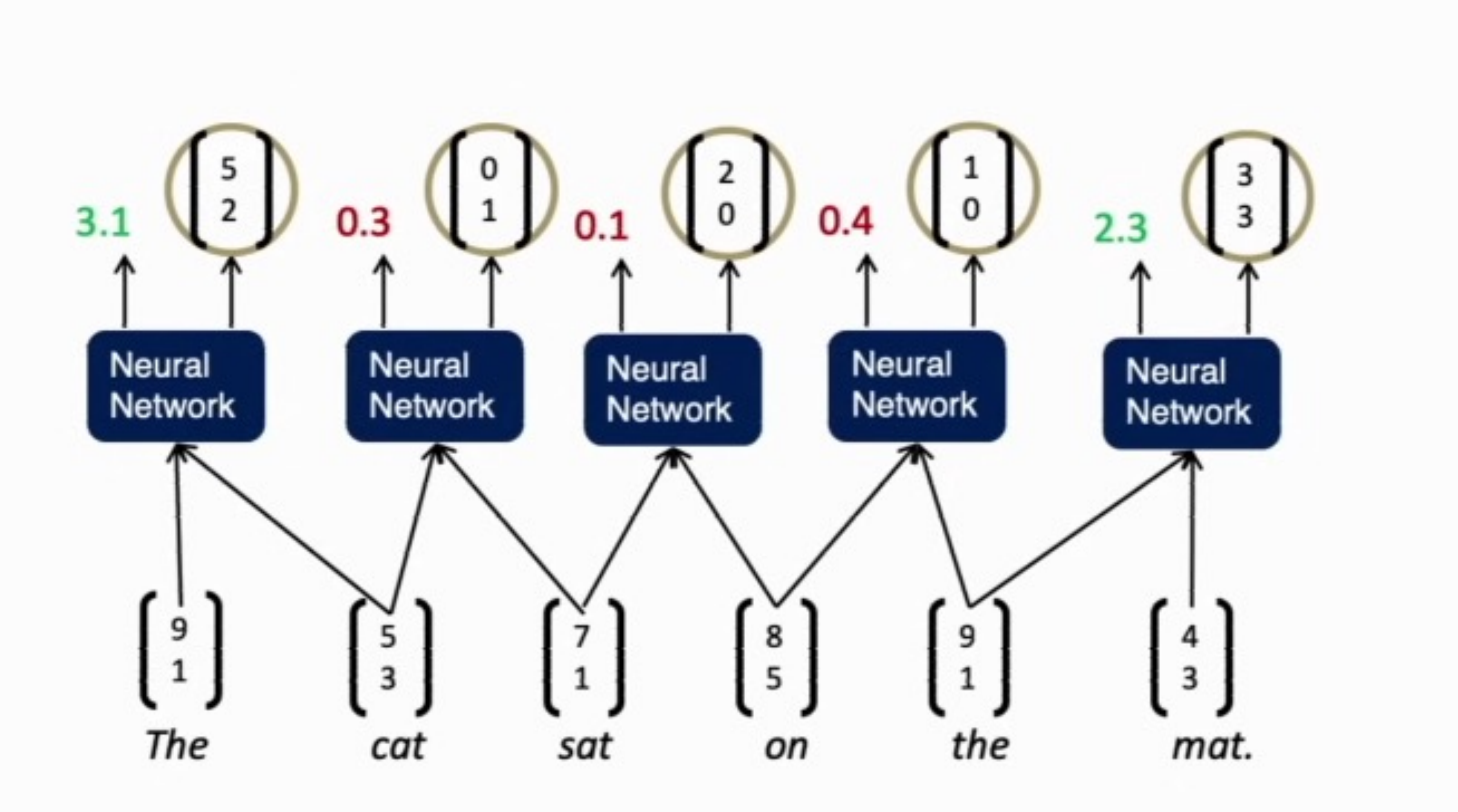
Parsing a sentence with an RNN

Greedy Method



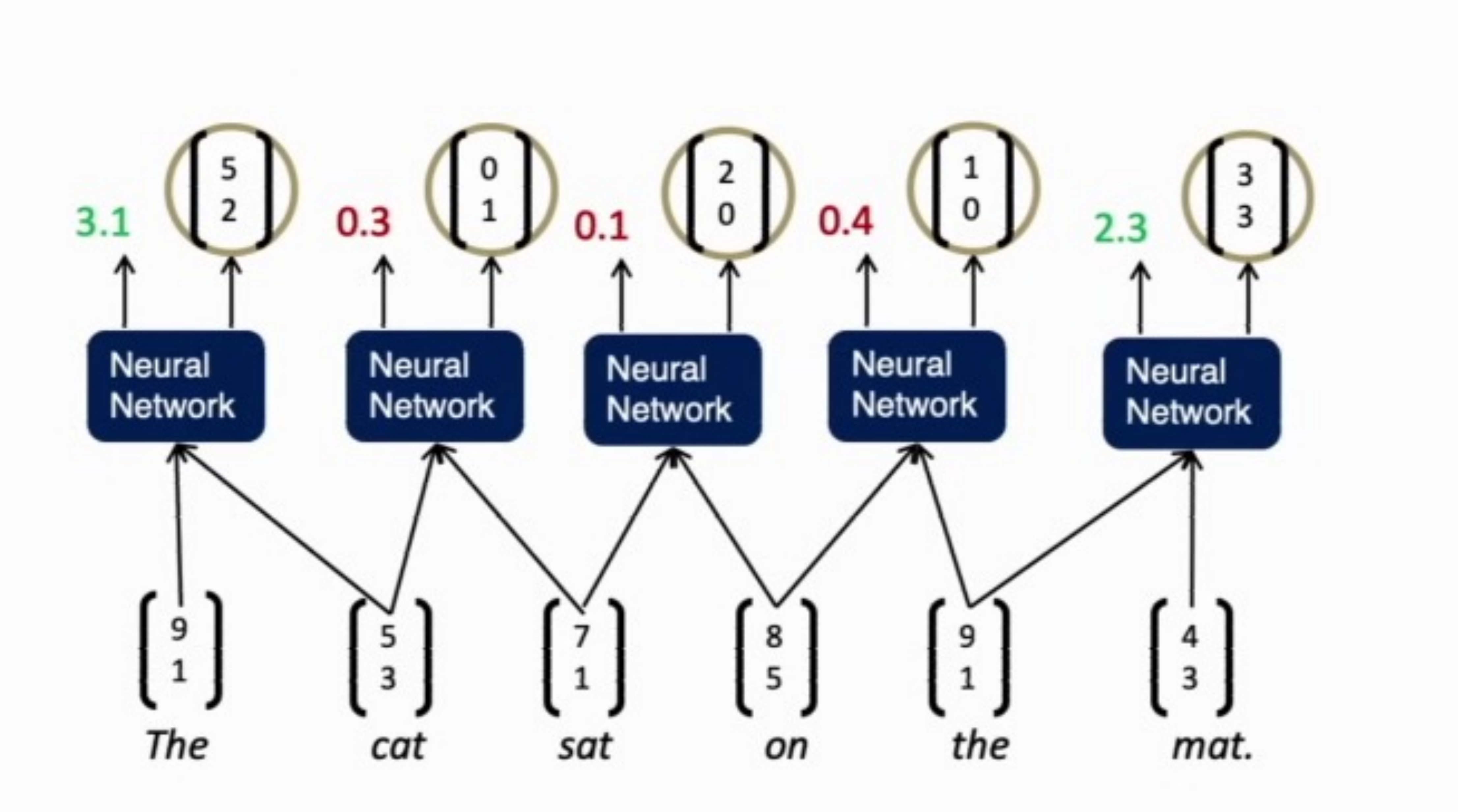
Parsing a sentence with an RNN

Greedy Method



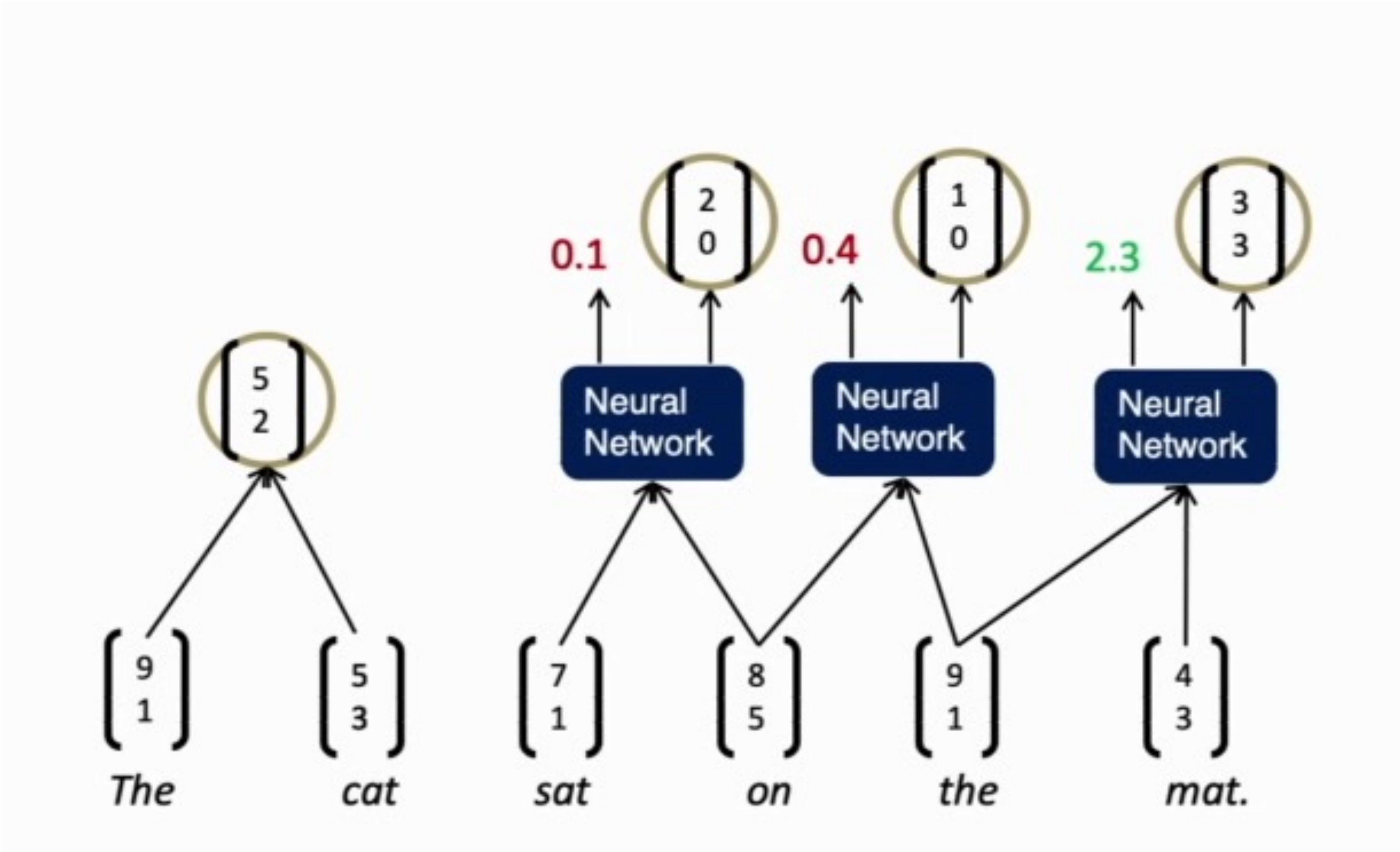
Parsing a sentence with an RNN

Greedy Method



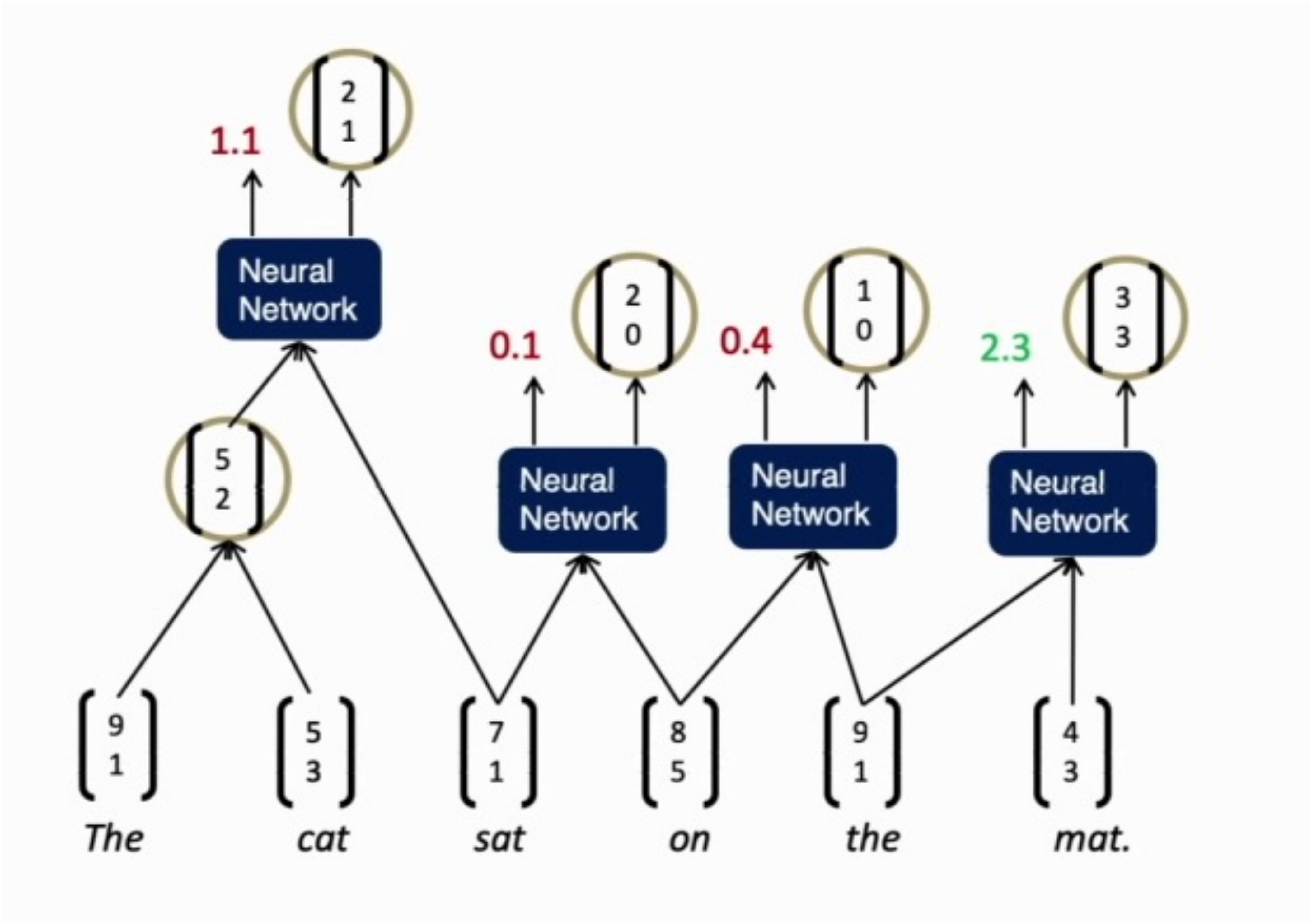
Parsing a sentence with an RNN

Greedy Method



Parsing a sentence with an RNN

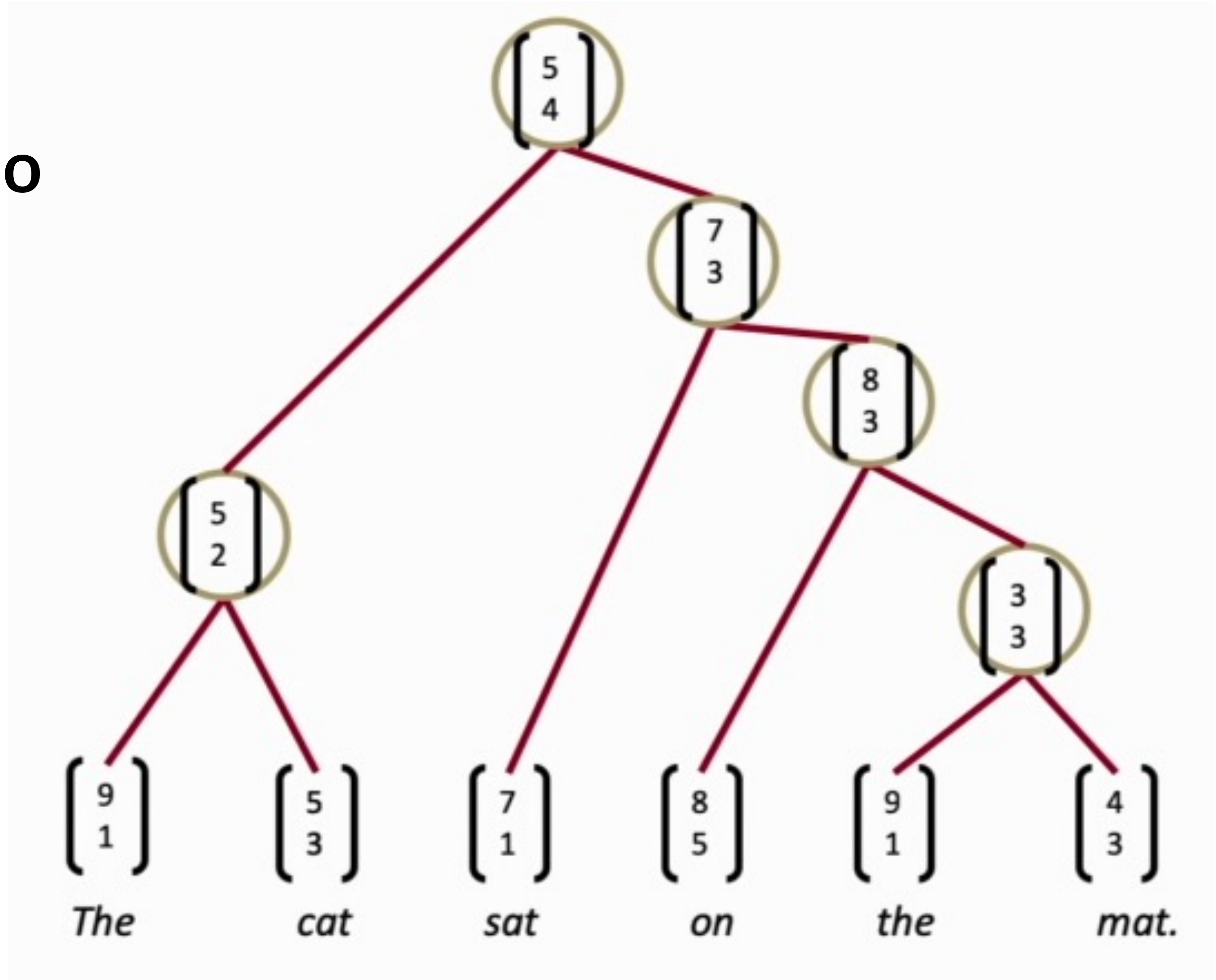
Greedy Method



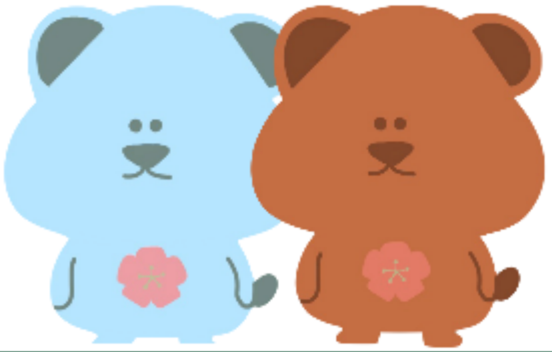
Parsing a sentence with an RNN

Greedy Method

- Beam Search can be applied too



BTS: Simple TreeRNN



BTS: Simple TreeRNN

Principally the same as general backpropagation

$$\boxed{\delta^{(l)} = \left((W^{(l)})^T \delta^{(l+1)}\right) \circ f'(z^{(l)}),} \quad \boxed{\frac{\partial}{\partial W^{(l)}} E_R = \delta^{(l+1)} (a^{(l)})^T + \lambda W^{(l)}}$$

- 1. 모든 노드의 W에 대한 derivative를 더함
- 2. 각 노드에서 derivative를 split
- 3. parent와 노드 자체에서 에러 메시지 더함

BTS: Simple TreeRNN

BTS: 1) Sum derivatives of all nodes

You can actually assume it's a different *W* at each node

Intuition via example:

$$
\begin{aligned}
& \frac{\partial}{\partial W} f(W(f(Wx)) \\
= \quad & f'(W(f(Wx)) \left(\left(\frac{\partial}{\partial W} W \right) f(Wx) + W \frac{\partial}{\partial W} f(Wx) \right) \\
= \quad & f'(W(f(Wx)) \left(f(Wx) + W f'(Wx) x \right)
\end{aligned}
$$

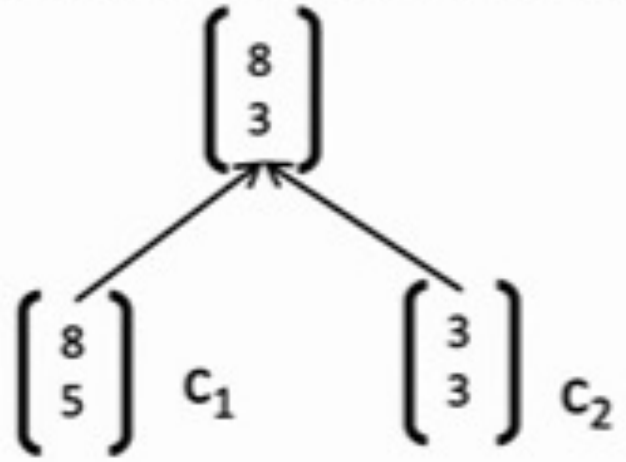
If we take separate derivatives of each occurrence, we get same:

$$
\begin{aligned}
& \frac{\partial}{\partial W_2} f(W_2(f(W_1 x)) + \frac{\partial}{\partial W_1} f(W_2(f(W_1 x)) \\
= \quad & f'(W_2(f(W_1 x)) \left(f(W_1 x) \right) + f'(W_2(f(W_1 x)) \left(W_2 f'(W_1 x) x \right) \\
= \quad & f'(W_2(f(W_1 x)) \left(f(W_1 x) + W_2 f'(W_1 x) x \right) \\
= \quad & f'(W(f(Wx)) \left(f(Wx) + W f'(Wx) x \right)
\end{aligned}
$$

BTS: Simple TreeRNN

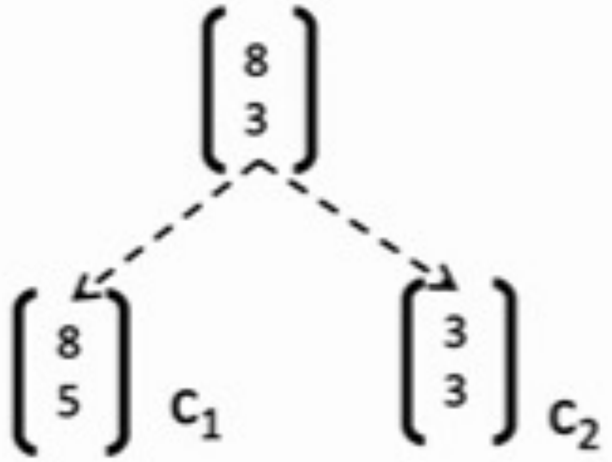
BTS: 2) Split derivatives at each node

During forward prop, the parent is computed using 2 children



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

Hence, the errors need to be computed wrt each of them:



where each child's error is *n*-dimensional

$$\delta_{p\to c_1 c_2} = [\delta_{p\to c_1} \delta_{p\to c_2}]$$

BTS: Simple TreeRNN

- Simple TreeRNN의 장점
  - 이전보다 더 큰 텍스트 단위의 의미 표현 가능

- Simple TreeRNN의 단점
  - 앞서 모든 노드에서 W가 동일하다고 설명했는데, 이는 더 복잡한 문장에서는 적절하지 못함
  - 인풋 단어 간 실제 상호작용이 없음
  - 조합 함수가 모든 경우에 대해 동일하게 작용

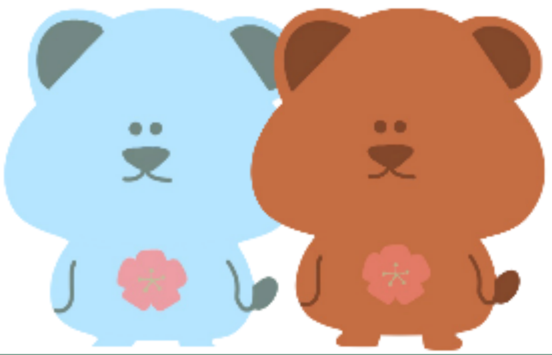
BTS: Simple TreeRNN

- Simple TreeRNN의 장점
  - 이전보다 더 큰 텍스트 단위의 의미 표현 가능

- Simple TreeRNN의 단점
  - 앞서 모든 노드에서 W가 동일하다고 설명했는데, 이는 더 복잡한 문장에서는 적절하지 못함
  - 인풋 단어 간 실제 상호작용이 없음
  - 조합 함수가 모든 경우에 대해 동일하게 작용

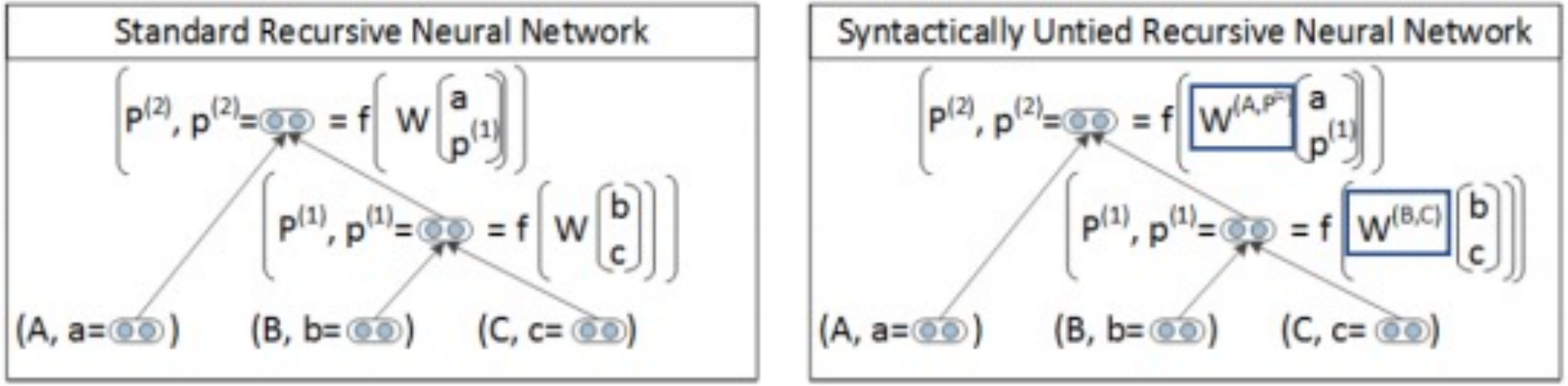
EWHA EURON

Syntactically-United RNN



Syntactically-United RNN

- 기능이 다른 표현에 각기 다른 가중치를 사용
- Simple TreeRNN 개선



Syntactically-United RNN

Compositional Vector Grammers

- 앞선 방식의 문제점: 속도가 느림. Greedy나 beam searc로 모든 score 후보군을 계산하는 것은 계산량이 많음

- 해결방법: Tree의 부분 집합에 대해서만 score 계산해서 빠르게 만듦 (PCFG)

Compositional Vector Grammer = PCFG + TreeRNN

Syntactically-United RNN

PCFG (Probabilistic Context Free Grammer)

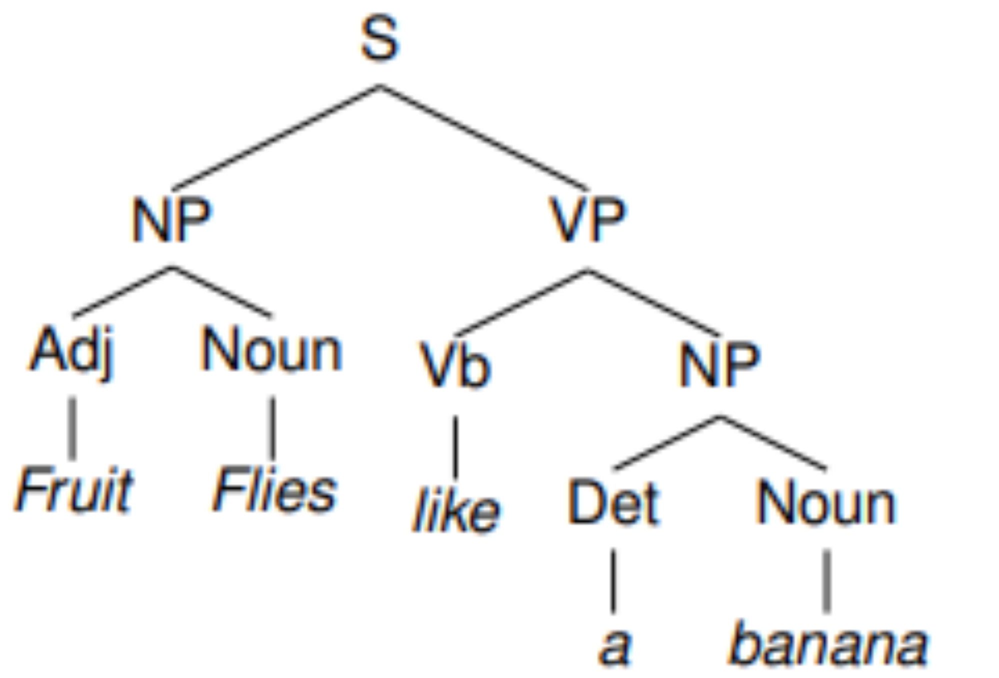
- 규칙에 따라 Weight matrix를 다르게 적용

PCFG Example

a simple PCFG

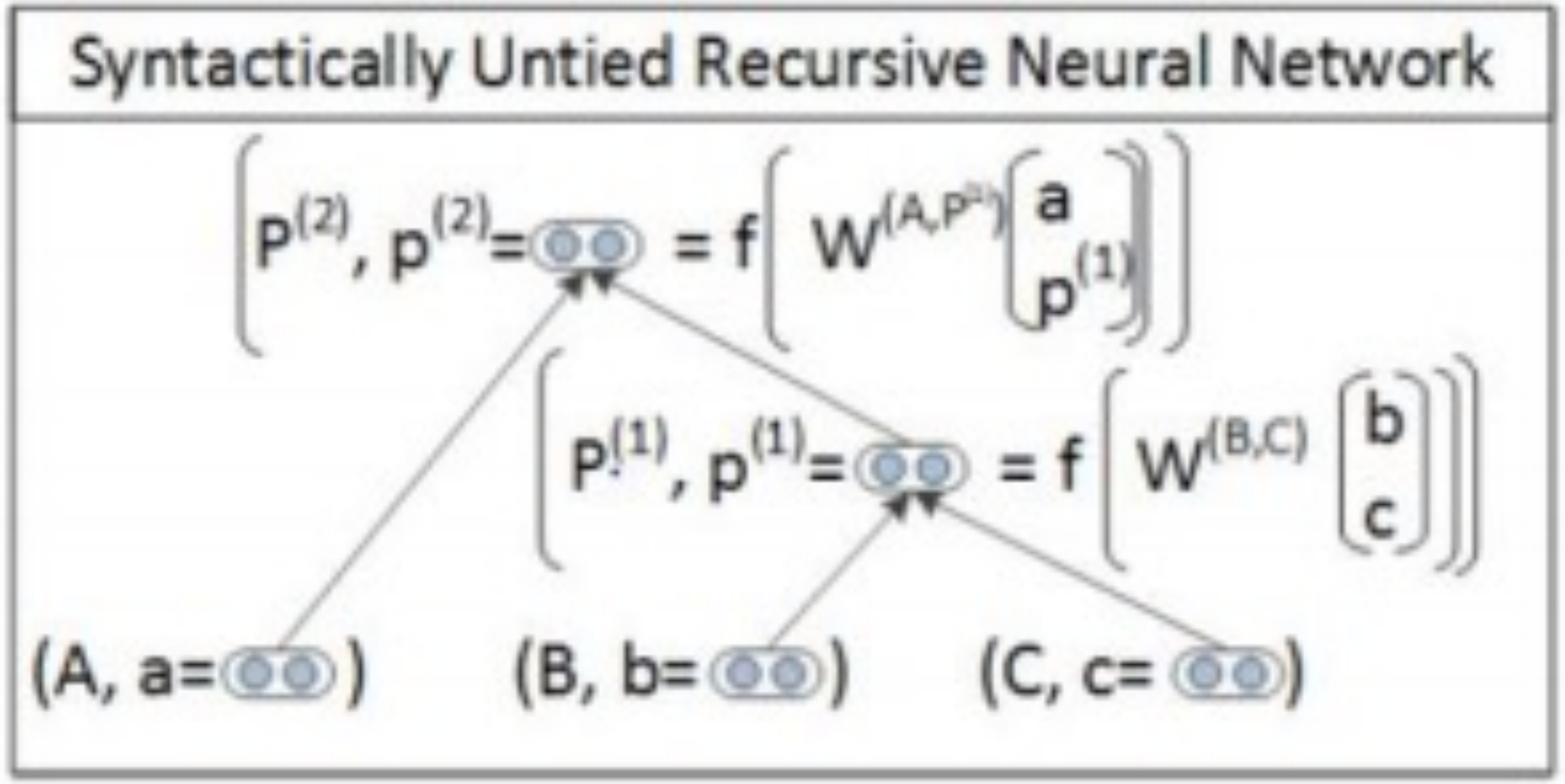
1.0 S → NP VP
0.3 NP → Adj Noun
0.7 NP → Det Noun
1.0 VP → Vb NP
-
0.2 Adj → fruit
0.2 Noun → flies
1.0 Vb → like
1.0 Det → a
0.4 Noun → banana
0.4 Noun → tomato
0.8 Adj → angry

Example



$1 * 0.3 * 0.2 * 0.7 * 1.0 * 0.2 * 1 * 1 * 0.4 = 0.0033$

Syntactically-United RNN



PCFG + TreeRNN

THANK YOU