

14기 NLP seminar

ToBig's 13기 오진석

# Lecture 5 - Dependency Parsing

# Contents

---

Unit 01 | Syntactic Structure: Constituency and Dependency

---

Unit 02 | Dependency Grammar and Treebanks

---

Unit 03 | Transition-based dependency parsing

---

Unit 04 | Neural dependency parsing

---

# Unit 01 | Syntactic Structure: Constituency and Dependency

## Unit 01 | Syntactic Structure

### Linguistic Structure, 문장 구조

#### Parsing

각 문장의 문법적인 구성 또는 구문을 분석하는 과정

#### Constituency Parsing

문장의 **구성요소**를 파악하여 구조를 분석하는 방법

#### Dependency Parsing

**단어간 의존 관계**를 파악하여 구조를 분석하는 방법

# Unit 01 | Syntactic Structure

## Constituency Parsing

: phrase-structure grammar 혹은 context-free grammar라고도 하며, 문장을 구성하고 있는 구(phrase)를 파악하여 문장 구조를 분석  
영어와 같이 어순이 비교적 고정적인 언어에서 주로 사용됨

the,	cat,	cuddly,	by,	door
Det	N	Adj	P	N

the cuddly cat,	by the door
NP -> Det + Adj + N	PP -> P + N

The cuddly cat by the door
NP -> NP + PP

# Unit 01 | Syntactic Structure

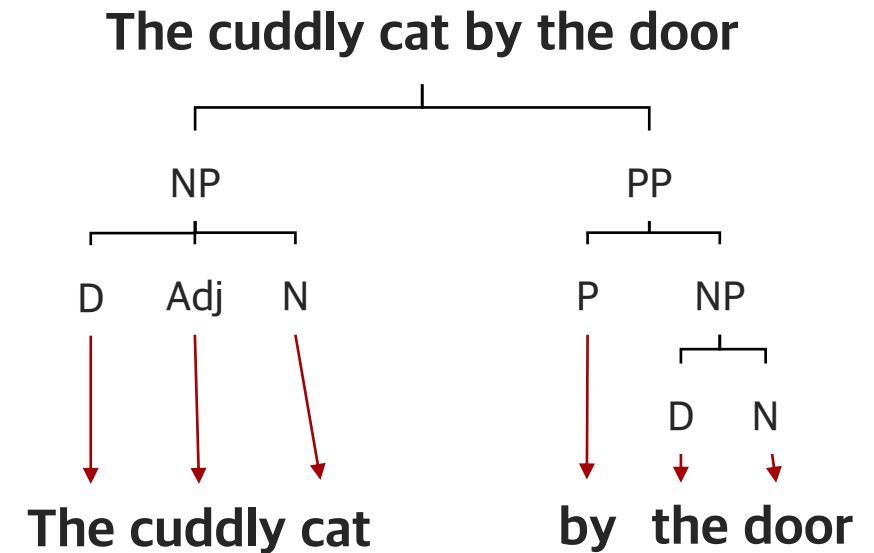
## Constituency Parsing

: phrase-structure grammar 혹은 context-free grammar라고도 하며, 문장을 구성하고 있는 구(phrase)를 파악하여 문장 구조를 분석  
영어와 같이 어순이 비교적 고정적인 언어에서 주로 사용됨

the,	cat,	cuddly,	by,	door
Det	N	Adj	P	N

the cuddly cat,	by the door
NP -> Det + Adj + N	PP -> P + N

The cuddly cat by the door
NP -> NP + PP



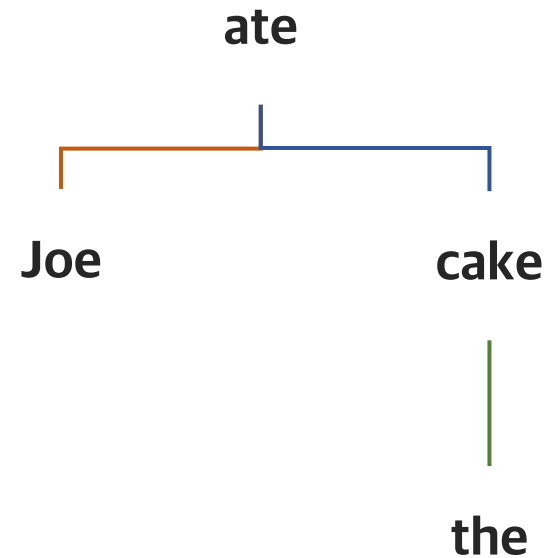
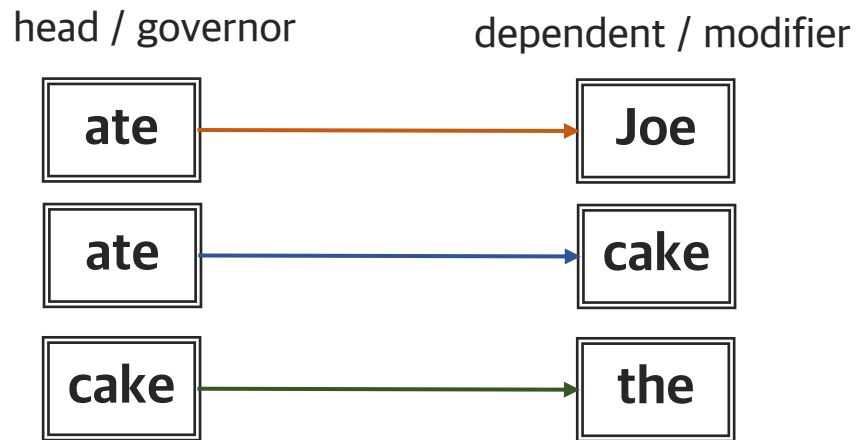
# Unit 01 | Syntactic Structure

## Dependency Parsing

: 문장에 존재하는 단어 간 의존 또는 수식 방향으로 관계를 파악하여 문장 구조를 분석

한국어와 같이 자유 어순을 가지거나 문장 성분이 생략 가능한 언어에서 선호, 최근에는 영어에서도 dependency parsing에 대한 관심이 증가

“ Joe ate the cake ”



## Unit 01 | Syntactic Structure

### Why do we need the sentence structure?

: 언어를 정확하게 해석하기 위해서 문장 구조에 대한 이해가 필요하며,

인간은 복잡한 단어의 구성을 통해 의사소통을 하기 때문에 정확한 이해를 위해서는 연결 구조에 대해서 알고 있어야함



## Unit 01 | Syntactic Structure

### Why do we need the sentence structure?

- Phrase Attachment Ambiguity

: 형용사구, 동사구, 전치사구 등이 어떤 단어를 수식하는지에 따라 의미가 달라지는 모호성

“ Scientists observe whales from space ”



Scientists observe whales from space

과학자들은 우주에서 고래를 관측했다.



Scientists observe whales from space


과학자들은 우주에서 온 고래를 관측했다.

## Unit 01 | Syntactic Structure


### Why do we need the sentence structure?

- Coordination Scope Ambiguity

: 특정 단어가 작용(수식)하는 대상의 범위가 달라짐에 따라 의미가 변하는 모호성



**[Shuttle veteran and longtime NASA executive] Fred Gregory appointed to board**  
우주선 베테랑이자 오랜 NASA의 임원인 Fred Gregory가 이사로 임명되었다.



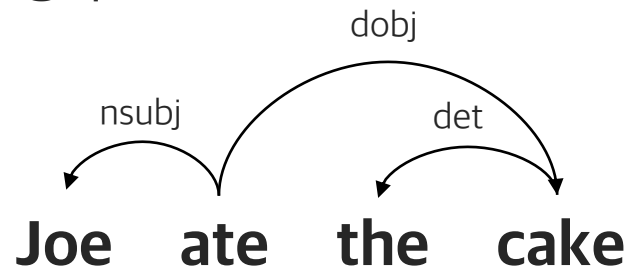
**Shuttle veteran and [longtime NASA executive] Fred Gregory appointed to board**  
우주선 베테랑과 오랜 NASA의 임원인 Fred Gregory가 이사로 임명되었다.

# Unit 02 | Dependency Grammar and Treebanks

## Unit 02 | Dependency Grammar and Treebanks

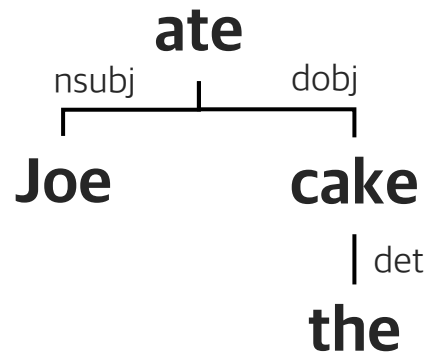
### Dependency structure

- sequence 형태



✓ Dependency structure는 두 가지 형태로 표현 가능

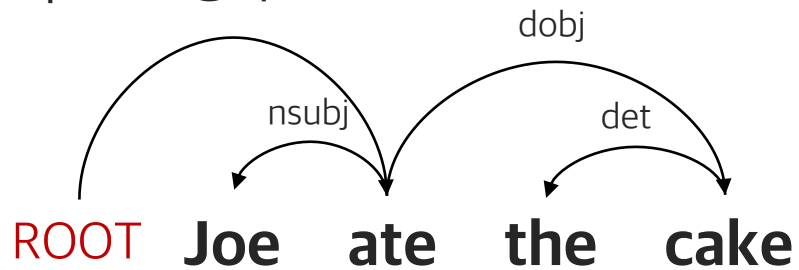
- tree 형태



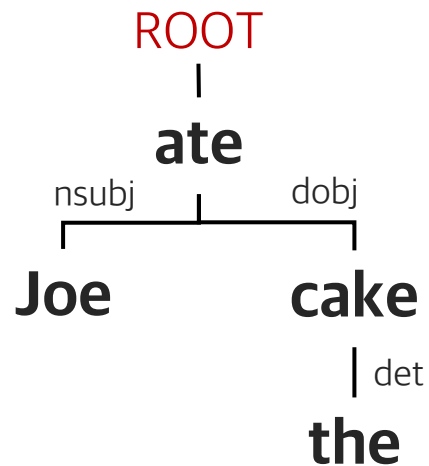
## Unit 02 | Dependency Grammar and Treebanks

### Dependency structure

- sequence 형태



- tree 형태

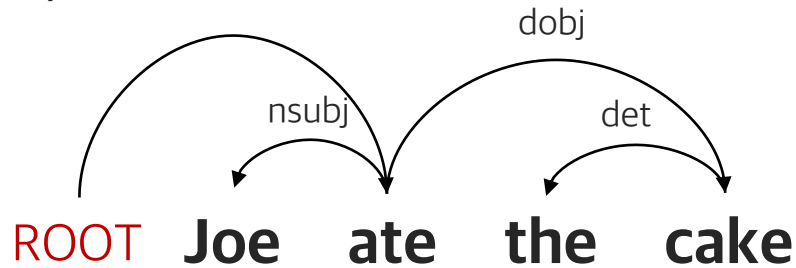


- ✓ Dependency structure는 **두 가지 형태**로 표현 가능
- ✓ 가상의 노드 ROOT를 추가하여 모든 성분의 **최종 head를 ROOT로** 설정
- ✓ ROOT는 모든 단어가 **최소 1개 노드의 dependent**가 되도록 함

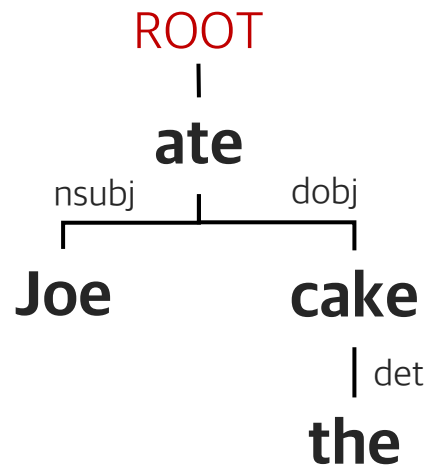
## Unit 02 | Dependency Grammar and Treebanks

### Dependency structure

- sequence 형태



- tree 형태

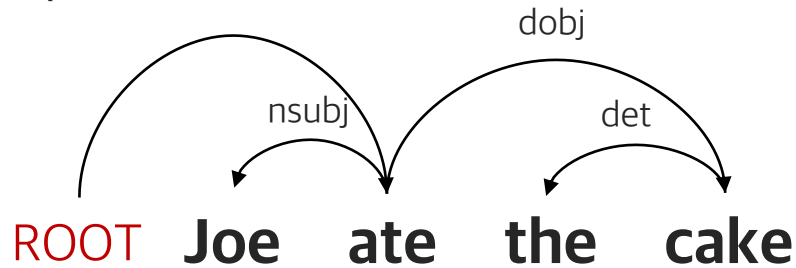


- ✓ Dependency structure는 **두 가지 형태**로 표현 가능
- ✓ 가상의 노드 ROOT를 추가하여 모든 성분의 **최종 head를 ROOT로** 설정
- ✓ ROOT는 모든 단어가 **최소 1개 노드의 dependent**가 되도록 함
- ✓ 화살표는 head에서 dependent로 함함
- ✓ 화살표 위 표시는 단어 간 **문법적 관계(dependency)**를 의미

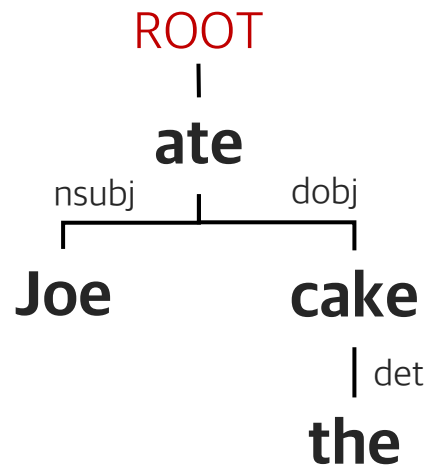
## Unit 02 | Dependency Grammar and Treebanks

### Dependency structure

- sequence 형태



- tree 형태



- ✓ Dependency structure는 **두 가지 형태**로 표현 가능
- ✓ 가상의 노드 **ROOT**를 추가하여 모든 성분의 **최종 head**를 **ROOT**로 설정
- ✓ **ROOT**는 모든 단어가 **최소 1개 노드의 dependent**가 되도록 함
- ✓ 화살표는 head에서 dependent로 향함
- ✓ 화살표 위 표시는 단어 간 **문법적 관계(dependency)**를 의미
- ✓ 화살표는 순환하지 않으며 중복의 관계가 형성되지 않음

## Unit 02 | Dependency Grammar and Treebanks

### Dependency Conditioning Preferences

: Dependency parsing의 보편적 특징

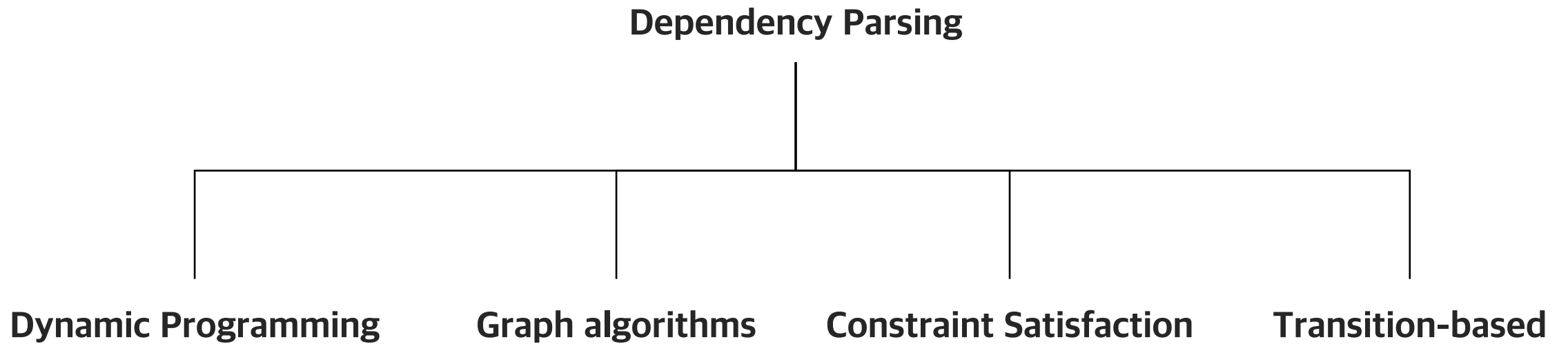
- **Bilexical affinities**: 두 단어 사이의 실제 의미가 드러나는 관계 ( discussions -> issues )
- **Dependency distance**: dependency의 거리를 의미하며, 주로 가까운 위치에서 dependent 관계가 형성
- **Intervening material**: 마침표, 세미클론과 같은 구두점을 넘어 dependent한 관계가 형성되지는 않음
- **Valency of heads**: head의 좌우측에 몇 개의 dependents를 가질 것인가에 대한 특성



# Unit 03 | Transition-based dependency parsing

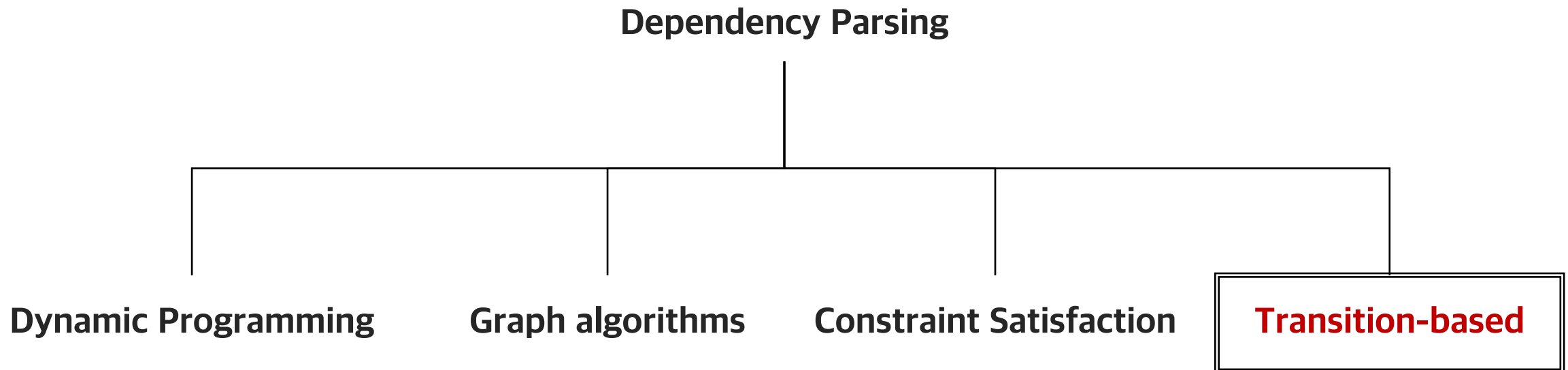
## Unit 03 | Transition-based dependency parsing

### Methods of Dependency Parsing



## Unit 03 | Transition-based dependency parsing

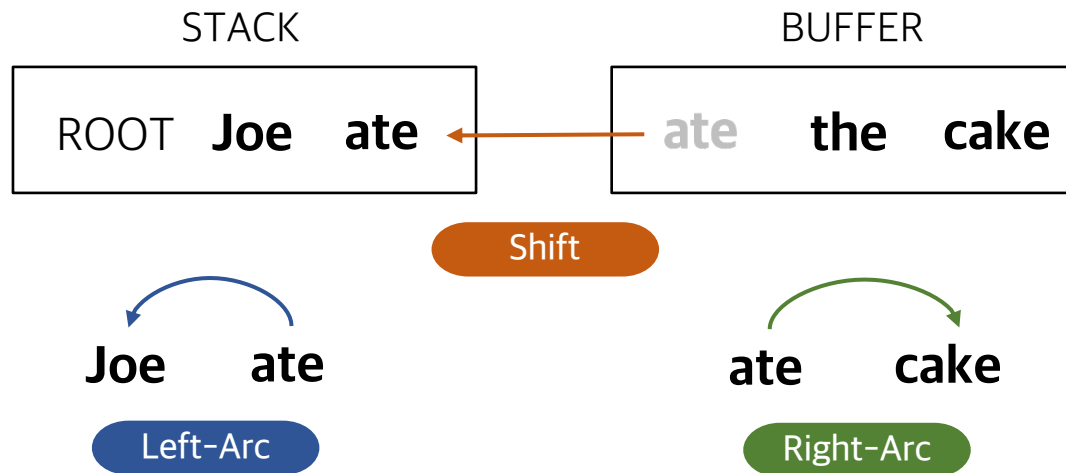
### Methods of Dependency Parsing



## Unit 03 | Transition-based dependency parsing

### Transition-based parsing

: 두 단어의 의존 여부를 차례대로 결정해나가면서 점진적으로 dependency structure를 구성



- ✓ Joakin Nivre(2003)에 의해 고안됨
- ✓ 각 상황(state)에서 Decision을 통해 parsing
- ✓ 단어 간 dependency를 classification하는 것과 유사함
- ✓ Chen and Manning(2014)에서 dense feature를 사용한 신경망 기반 parser를 제안하여 속도와 성능 모두 향상 시킴

## Unit 03 | Transition-based dependency parsing

### Transition-based parsing

Joe ate the cake

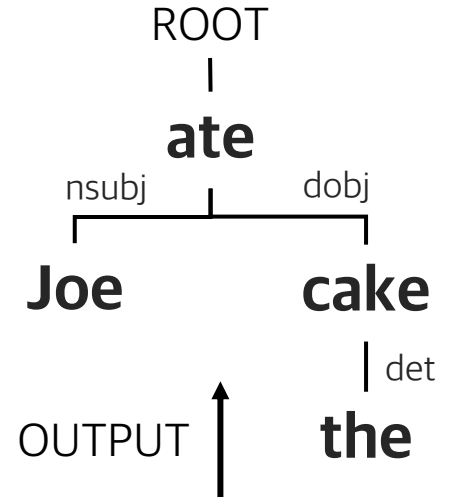
↓ INPUT

Joe ate the cake

BUFFER

ROOT [ State (c) ]

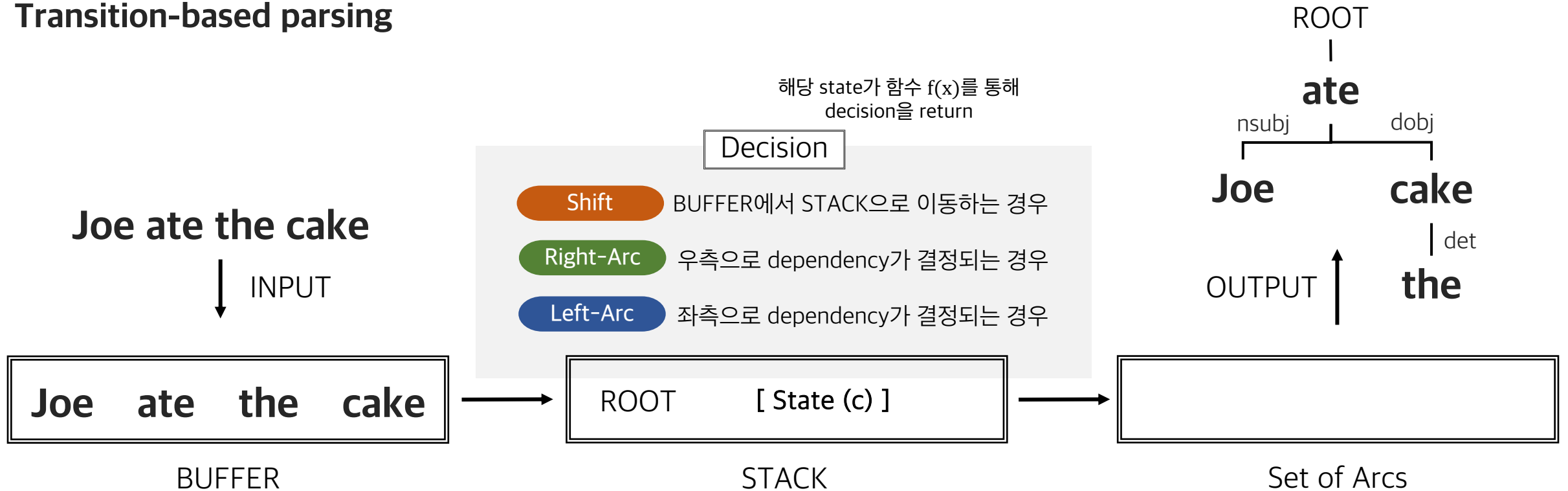
STACK



Set of Arcs

## Unit 03 | Transition-based dependency parsing

### Transition-based parsing

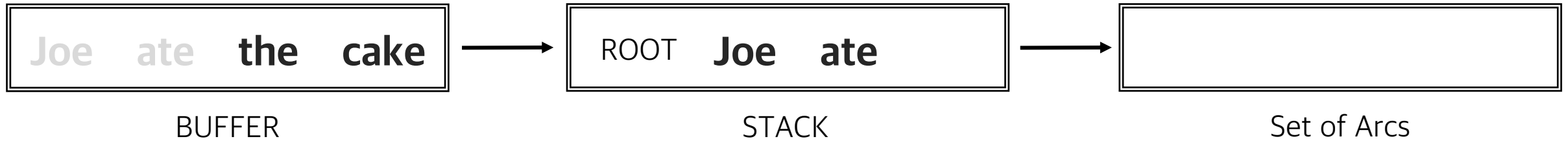


## Unit 03 | Transition-based dependency parsing

### Transition-based parsing

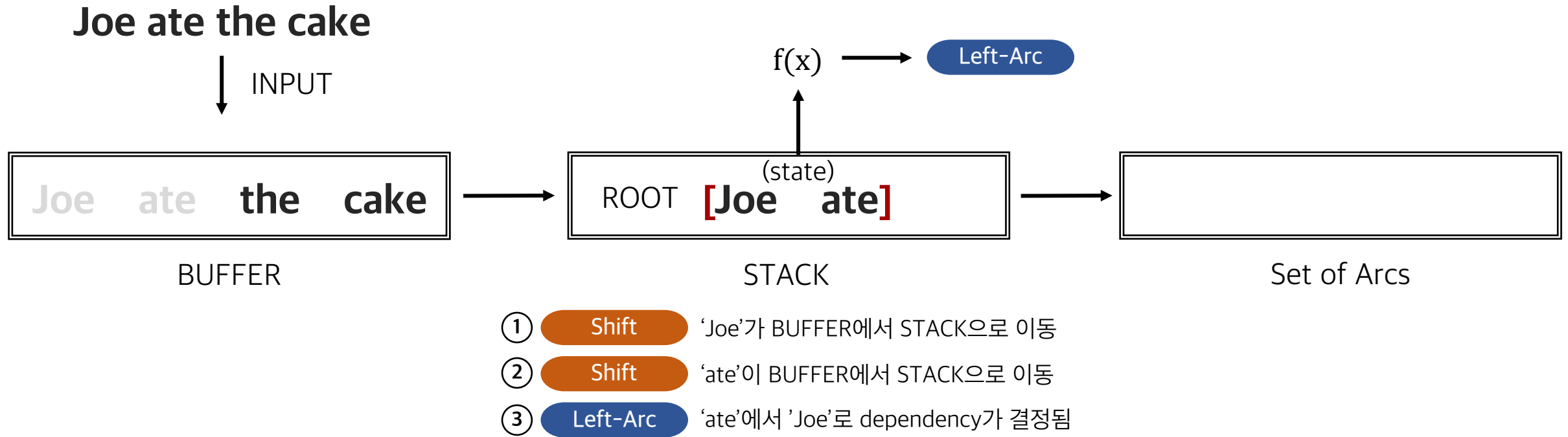
Joe ate the cake

↓ INPUT



## Unit 03 | Transition-based dependency parsing

### Transition-based parsing





## Unit 03 | Transition-based dependency parsing

### Transition-based parsing

Joe ate the cake

↓ INPUT



- ① **Shift** 'Joe'가 BUFFER에서 STACK으로 이동
- ② **Shift** 'ate'이 BUFFER에서 STACK으로 이동
- ③ **Left-Arc** 'ate'에서 'Joe'로 dependency가 결정됨
- ④ **Shift** 'the'가 BUFFER에서 STACK으로 이동

## Unit 03 | Transition-based dependency parsing

### Transition-based parsing

Joe ate the cake

↓  
INPUT

Joe ate the cake

BUFFER

ROOT

STACK

- |   |          |   |           |
|---|----------|---|-----------|
| ① | Shift    | ⑤ | Shift     |
| ② | Shift    | ⑥ | Left-Arc  |
| ③ | Left-Arc | ⑦ | Right-Arc |
| ④ | Shift    | ⑧ | Right-Arc |

Set of Arcs

(ate, nsubj, Joe)	(cake, det, the)
(ate, dobj, cake)	(Root, root, ate)

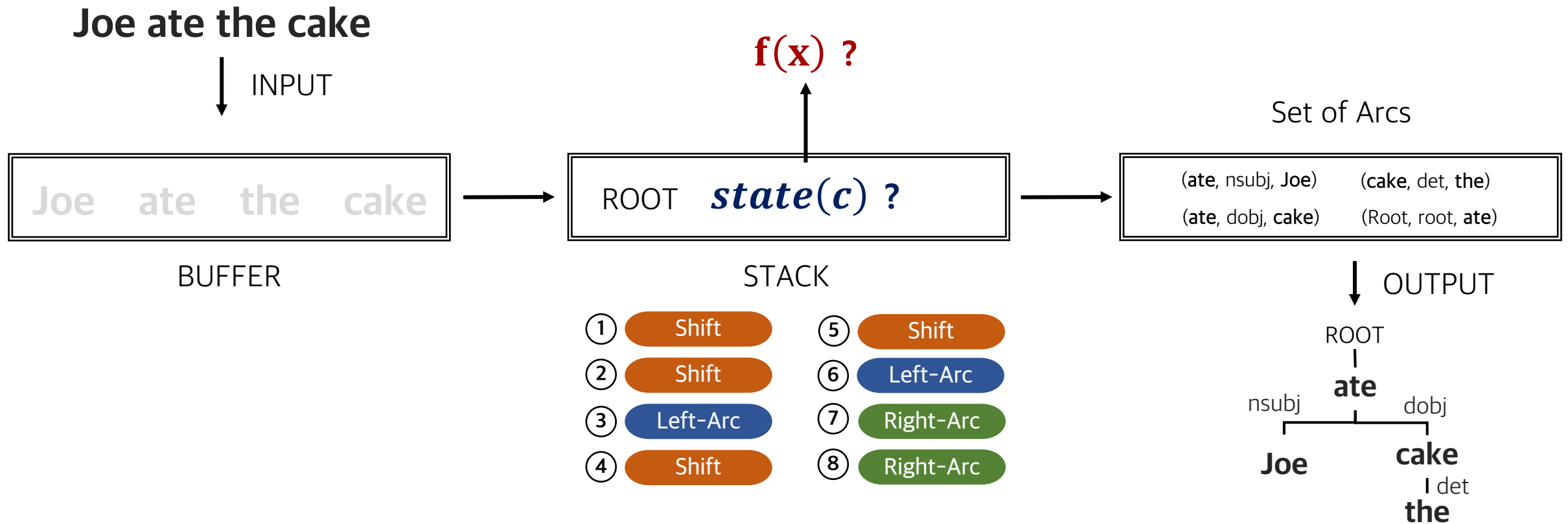
↓  
OUTPUT

```
graph TD
    ROOT[ROOT] --> ate[ate]
    ate -- nsubj --> Joe[Joe]
    ate -- dobj --> cake[cake]
    cake -- det --> the[the]
```

Joe ate the cake

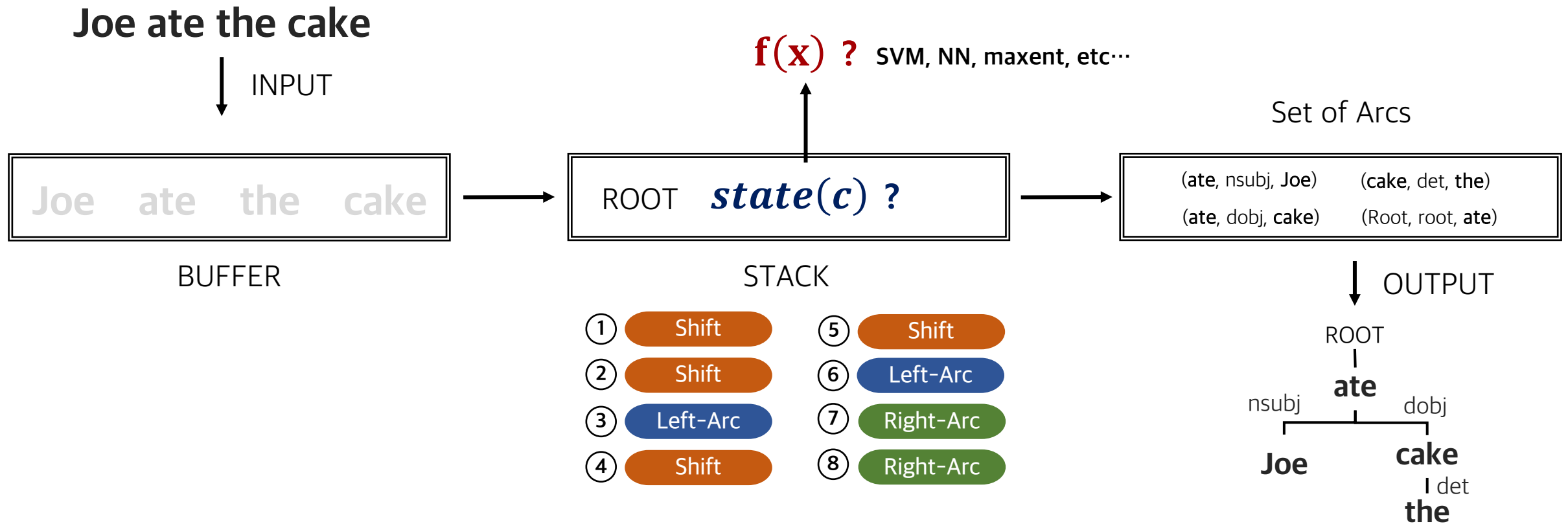
## Unit 03 | Transition-based dependency parsing

### Transition-based parsing



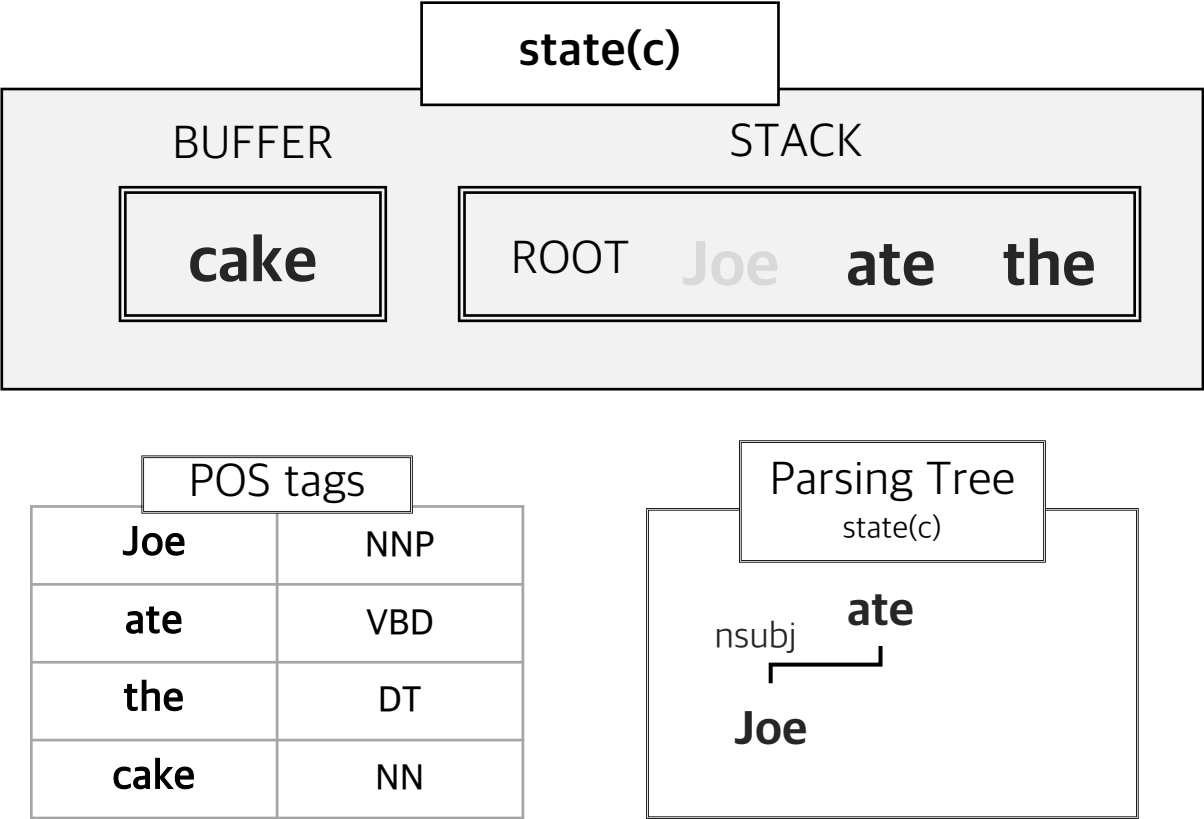
## Unit 03 | Transition-based dependency parsing

### Transition-based parsing



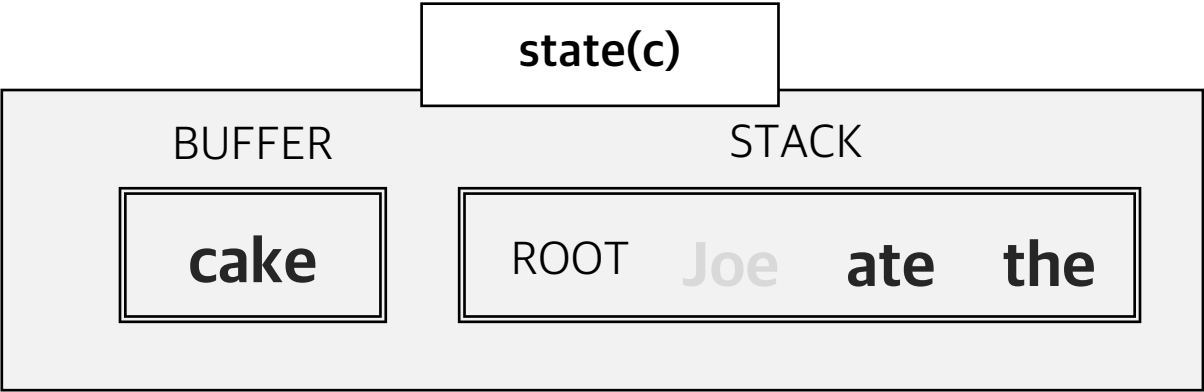
# Unit 03 | Transition-based dependency parsing

## MaltParser - Conventional Feature Representation

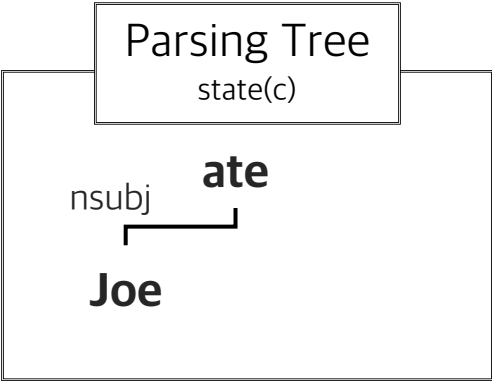


# Unit 03 | Transition-based dependency parsing

## MaltParser - Conventional Feature Representation



POS tags	
Joe	NNP
ate	VBD
the	DT
cake	NN

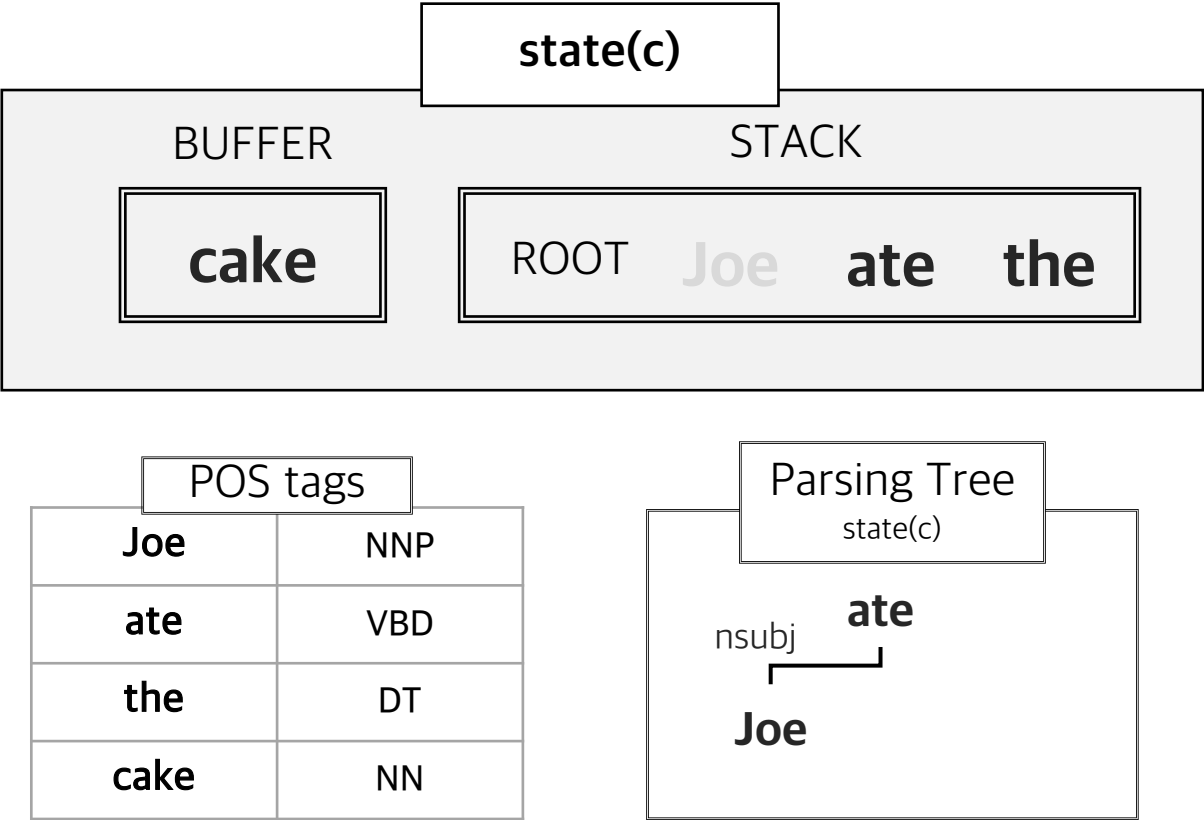


notations

- $s_1 \cdot W$  : STACK의 첫번째 단어
- $b_1 \cdot W$  : BUFFER의 첫번째 단어
- $s_1 \cdot t$  : STACK 첫번째 단어의 POS tag
- $lc(s_1) \cdot W$  : STACK 첫번째 단어의 left-child 단어
- $rc(s_1) \cdot W$  : STACK 첫번째 단어의 right-child 단어
- $lc(s_1) \cdot t$  : STACK 첫번째 단어의 left-child 단어의 POS tag

# Unit 03 | Transition-based dependency parsing

## MaltParser - Conventional Feature Representation



notations

- $s_1 \cdot W$  : STACK의 첫번째 단어 **the**
- $b_1 \cdot W$  : BUFFER의 첫번째 단어 **cake**
- $s_1 \cdot t$  : STACK 첫번째 단어의 POS tag **DT**
- $lc(s_2) \cdot W$  : STACK 두번째 단어의 left-child 단어 **Joe**
- $rc(s_1) \cdot W$  : STACK 첫번째 단어의 right-child 단어 **NULL**
- $lc(s_1) \cdot t$  : STACK 첫번째 단어의 left-child 단어의 POS tag **NULL**

Diagram illustrating a simple parser state:

- state(c)** (Current state)
- BUFFER**: **cake**
- STACK**: **ROOT** **Joe** **ate** **the**

The diagram illustrates a parsing tree for the sentence "Joe ate the apple". The root node is "Parsing Tree", which branches into "state(c)" and another node. The "state(c)" node branches into "nsobj" and "ate". The "nsobj" node branches into "Joe". The "ate" node branches into "the" and "apple".

state(c)

1

0

0

1

차원  $10^6 \sim 10^7$ 

•  
•  
•



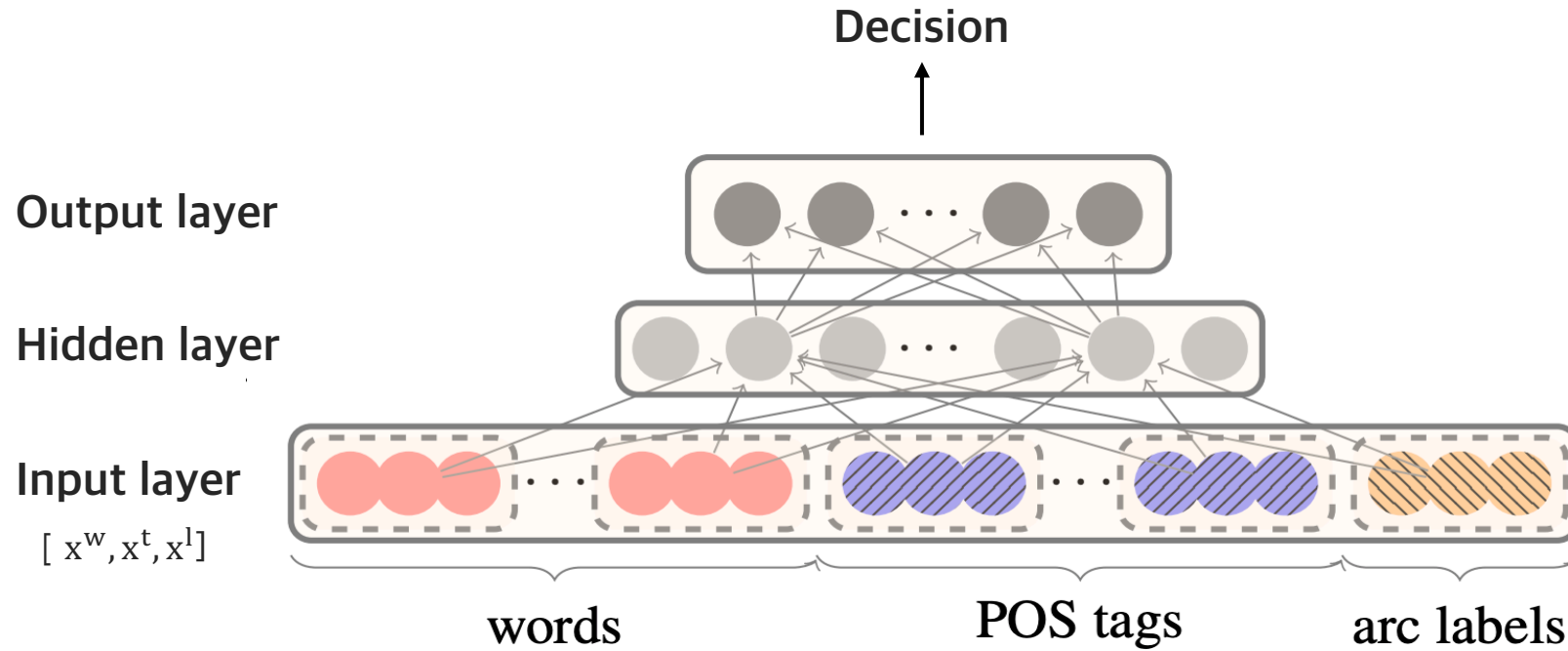


## Unit 04 | Neural dependency parsing

## Unit 04 | Neural dependency parsing

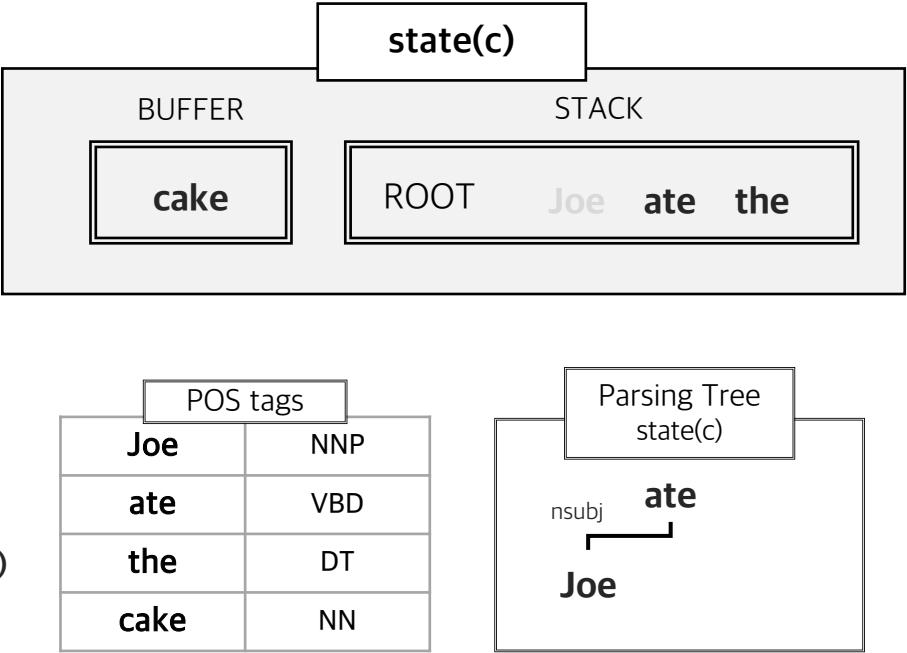
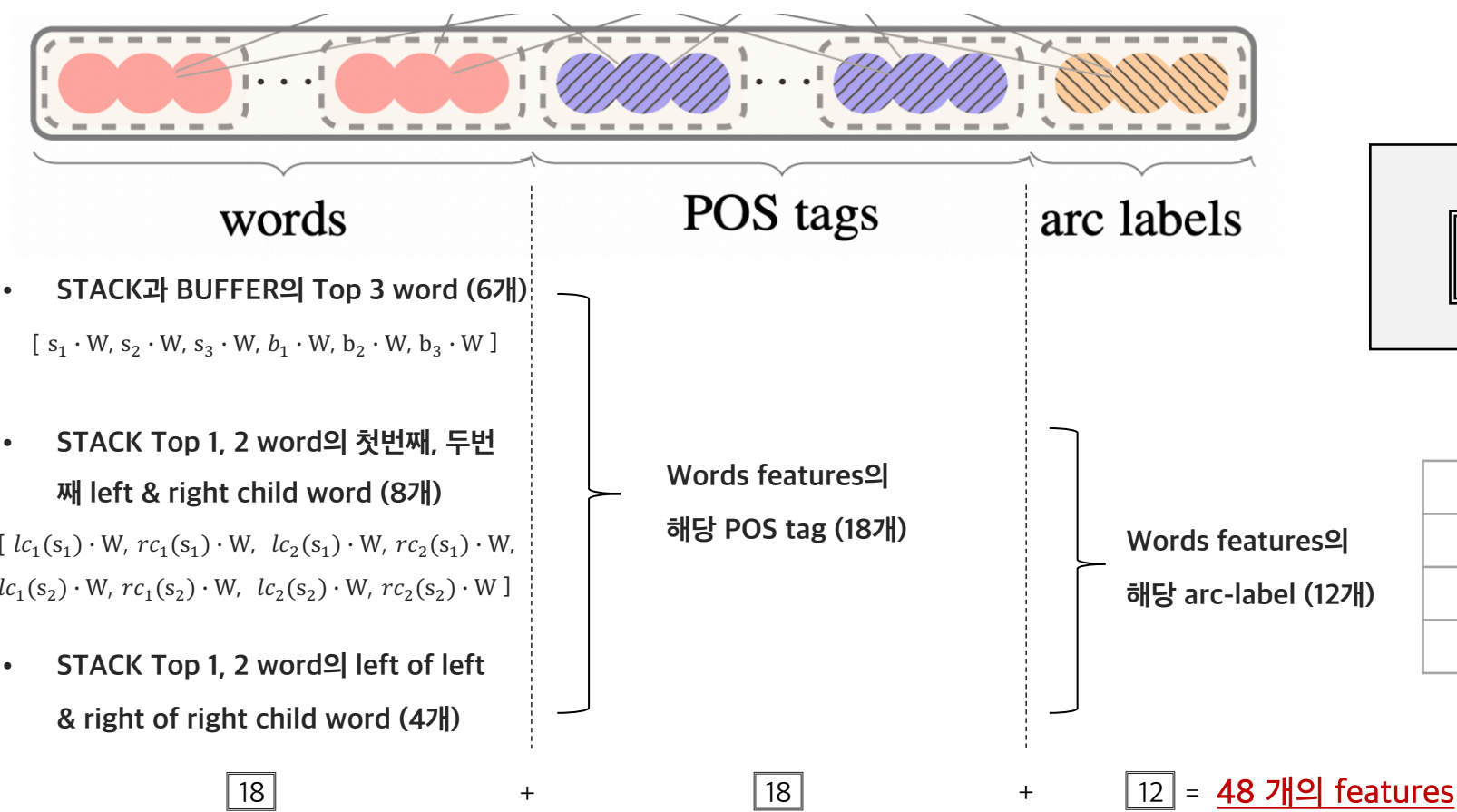
### Neural Dependency Parser

: 2014년에 발표된, Neural Net을 사용하여 dependency를 구하는 방법



# Unit 04 | Neural dependency parsing

## Neural Dependency Parser - Input layer



# Unit 04 | Neural dependency parsing

## Neural Dependency Parser - Input layer

words

[ cake, Null, Null, the, ate, ROOT,  
Null, Null, Null, Null, Joe, Null, Null, Null,  
Null, Null, Null, Null ]

One-hot

단어의 개수

	Null	ROOT	NNP	VBD	DT	NN
cake	0	0	0	0	0	1
Null	1	0	0	0	0	0
...	...					
ate	0	0	0	1	0	0

18

POS tag

[ NN, Null, Null, the, VBD, ROOT,  
Null, Null, Null, Null, NNP, Null, Null, Null,  
Null, Null, Null, Null ]

One-hot

POS tag 개수

	Null	ROOT	NNP	VBD	DT	NN
NN	0	0	0	0	0	1
Null	1	0	0	0	0	0
...	...					
VBD	0	0	0	1	0	0

18

Arc-label

[ Null, Null, Null, Null, nsubj, Null, Null, Null,  
Null, Null, Null, Null ]

One-hot

Arc-label 개수

	Null	ROOT	dobj	nsubj	det	...
Null	1	0	0	0	0	0
Null	1	0	0	0	0	0
...	...					
nsubj	0	0	0	1	0	0

12

# Unit 04 | Neural dependency parsing

## Neural Dependency Parser - Input layer

words

[ cake, Null, Null, the, ate, ROOT,  
Null, Null, Null, Null, Joe, Null, Null, Null,  
Null, Null, Null, Null ]

단어의 개수

	Null	ROOT	NNP	VBD	DT	NN
cake	0.7	1.0	1.1	0.2	0.6	1.3
Null	0.2	0.1	0.7	0.7	1.2	0.1
...	...					
ate	1.2	0.8	0.2	2.0	0.3	0.6

18

POS tag

[ NN, Null, Null, the, VBD, ROOT,  
Null, Null, Null, Null, NNP, Null, Null, Null,  
Null, Null, Null, Null ]

POS tag 개수

	Null	ROOT	NNP	VBD	DT	NN
NN	0.6	1.2	0.1	0.8	0.3	0.2
Null	1.1	0.2	0.6	1.8	1.1	0.8
...	...					
VBD	1.1	0.5	1.2	1.0	0.2	0.1

18

Arc-label

[ Null, Null, Null, Null, nsubj, Null, Null, Null,  
Null, Null, Null, Null ]

Arc-label 개수

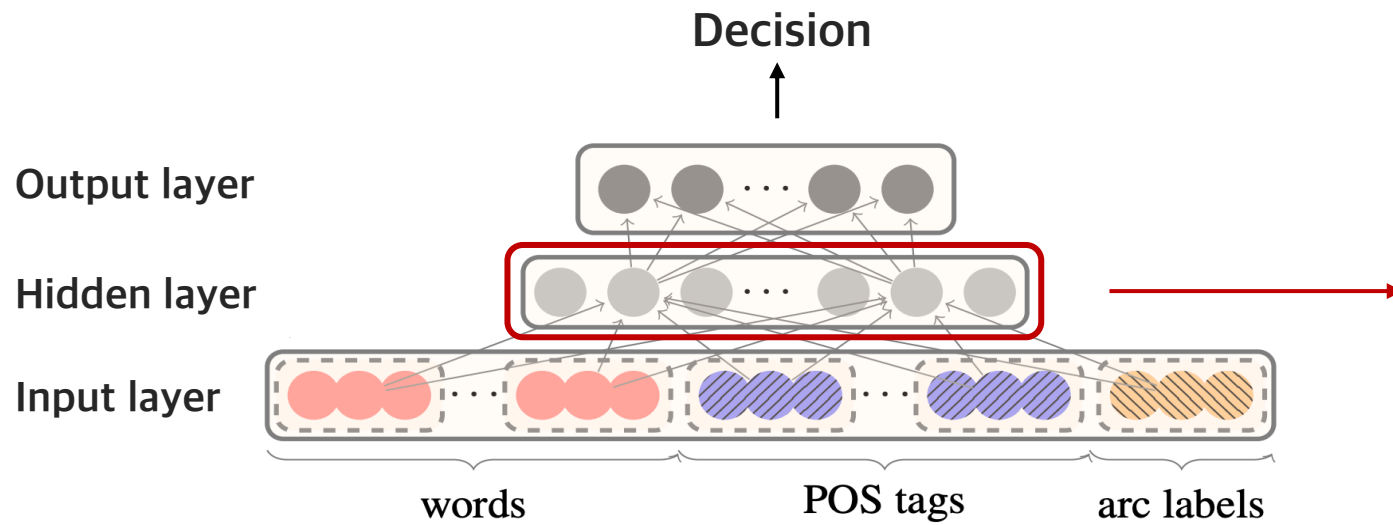
	Null	ROOT	dobj	nsubj	det	...
Null	0.8	0.1	0.6	0.2	1.4	0.8
Null	0.8	0.1	0.6	0.2	1.4	0.8
...	...					
nsubj	1.1	0.1	0.6	0.1	0.2	0.6

12



## Unit 04 | Neural dependency parsing

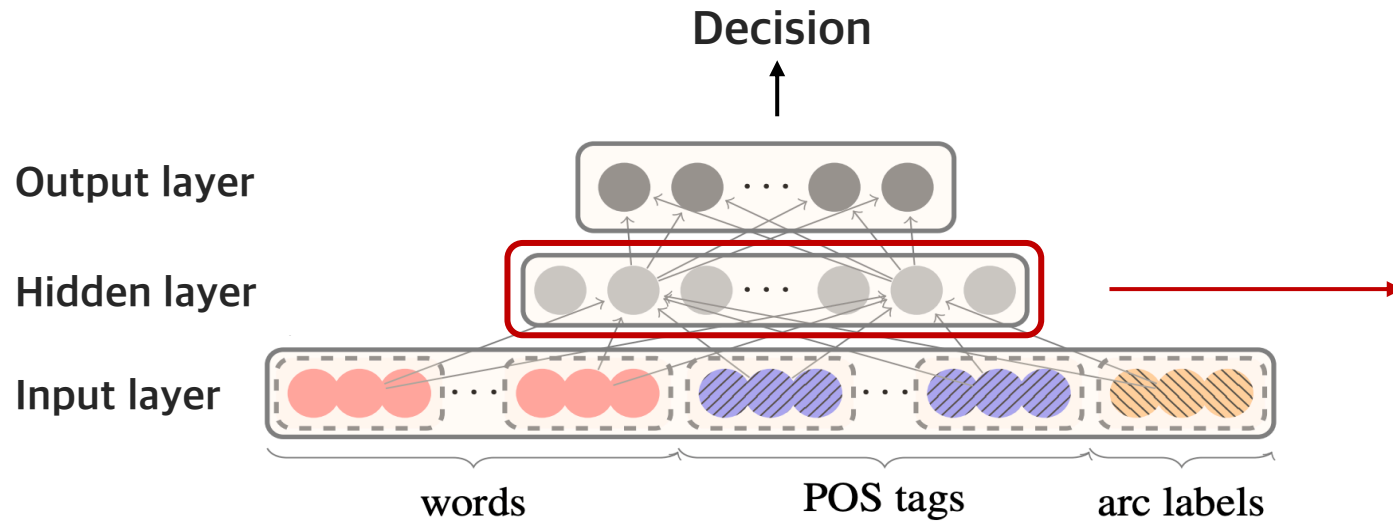
### Neural Dependency Parser - Hidden layer



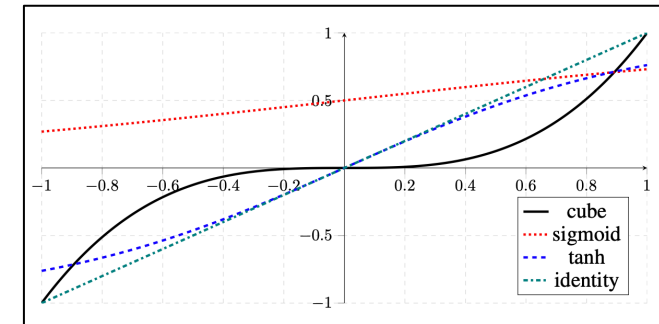


# Unit 04 | Neural dependency parsing

## Neural Dependency Parser - Hidden layer



Activation function - cube function

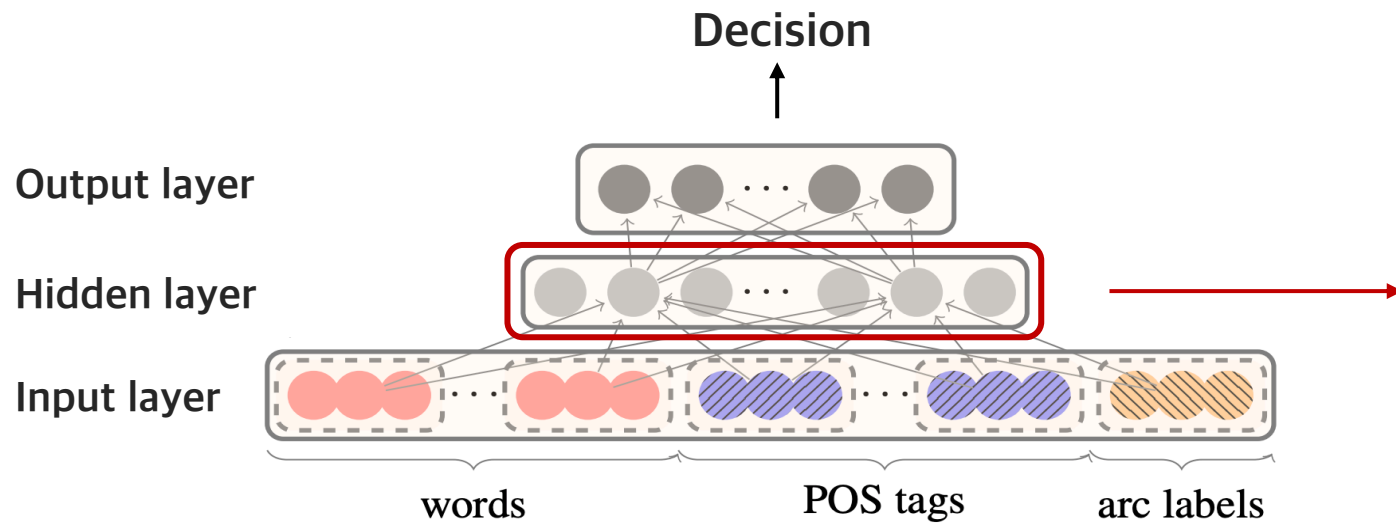


$$g(x) = x^3$$

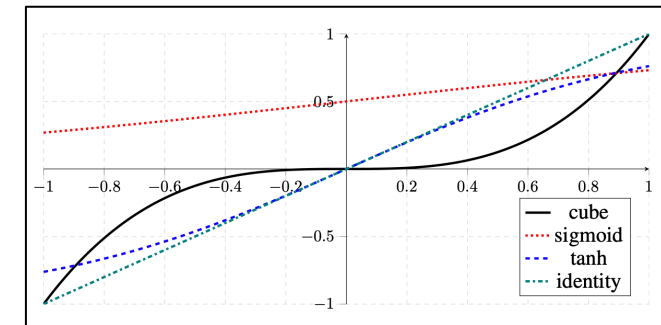
$$g(w_1x_1 + \dots + w_mx_m + b) = \sum_{i,j,k} (w_iw_jw_k)x_ix_jx_k + \sum_{i,j} b(w_iw_j)x_ix_j \dots$$

# Unit 04 | Neural dependency parsing

## Neural Dependency Parser - Hidden layer



Activation function - cube function



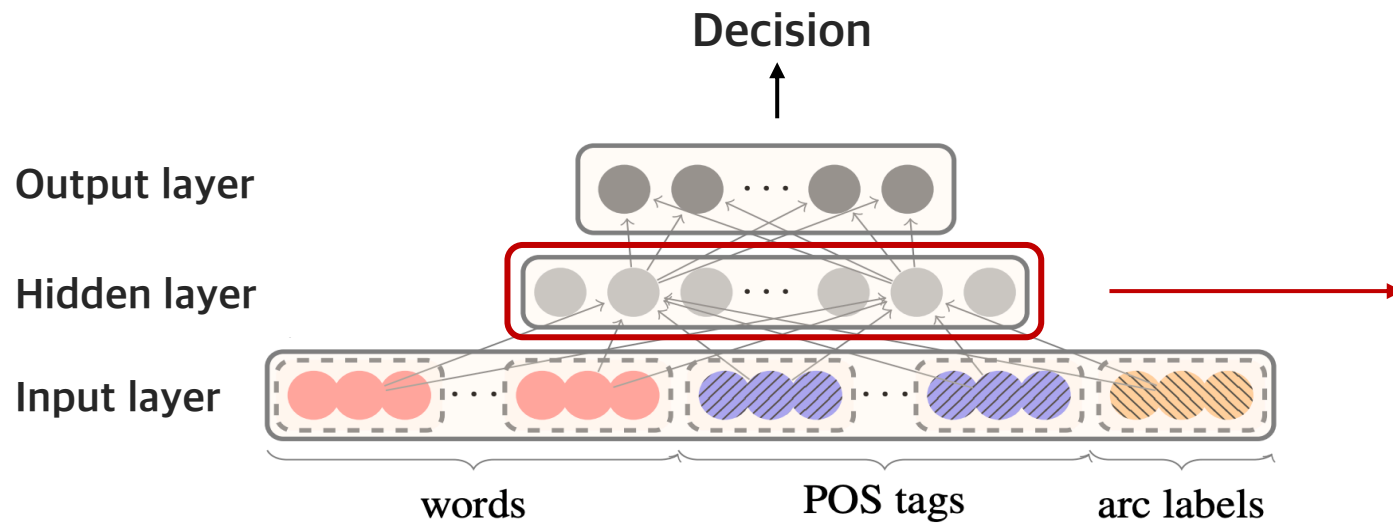
$$g(x) = x^3$$

$$g(w_1x_1 + \dots + w_mx_m + b) = \sum_{i,j,k} (w_iw_jw_k) x_ix_jx_k + \sum_{i,j} b(w_iw_j)x_ix_j \dots$$

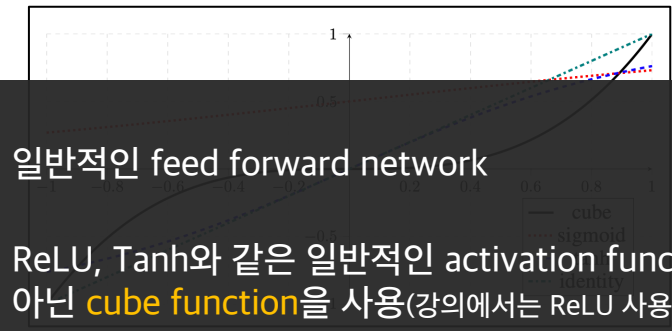
Words, POS tag, Arc-label간 상호작용을 반영할 수 있는 조합

## Unit 04 | Neural dependency parsing

## Neural Dependency Parser - Hidden layer



## Activation function - cube function

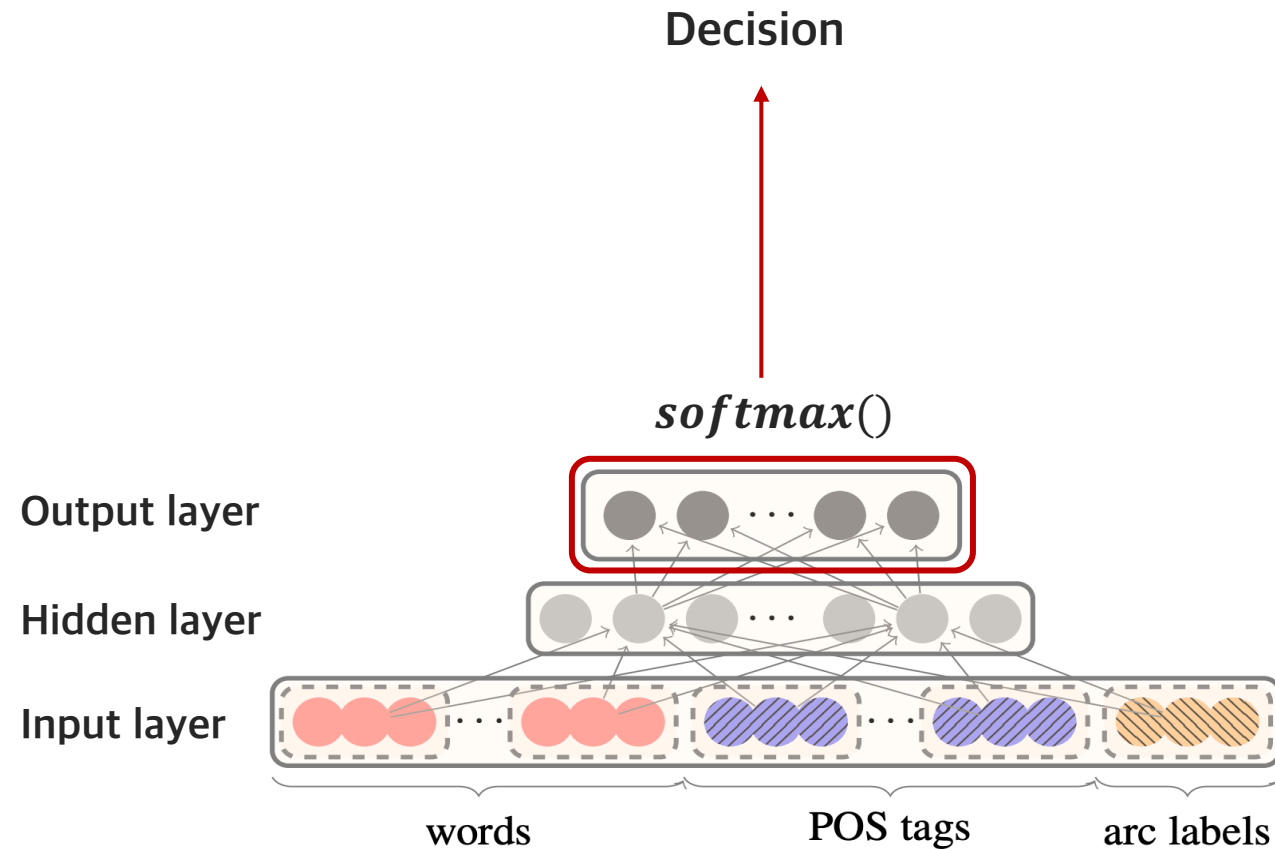


- 일반적인 feed forward network
- ReLU, Tanh와 같은 일반적인 activation function이 아닌 **cube function**을 사용(강의에서는 ReLU 사용)
- cube function은 words, pos tags, arc-label **feature** 간 상호작용을 반영할 수 있음
- 엄밀한 수학적 증명은 하지 않았으나, 실험 결과 타 **Non-linearity 대비 성능이 좋음**

Words, POS tag, Arc-label간  
상호작용을 반영할 수 있는 조합

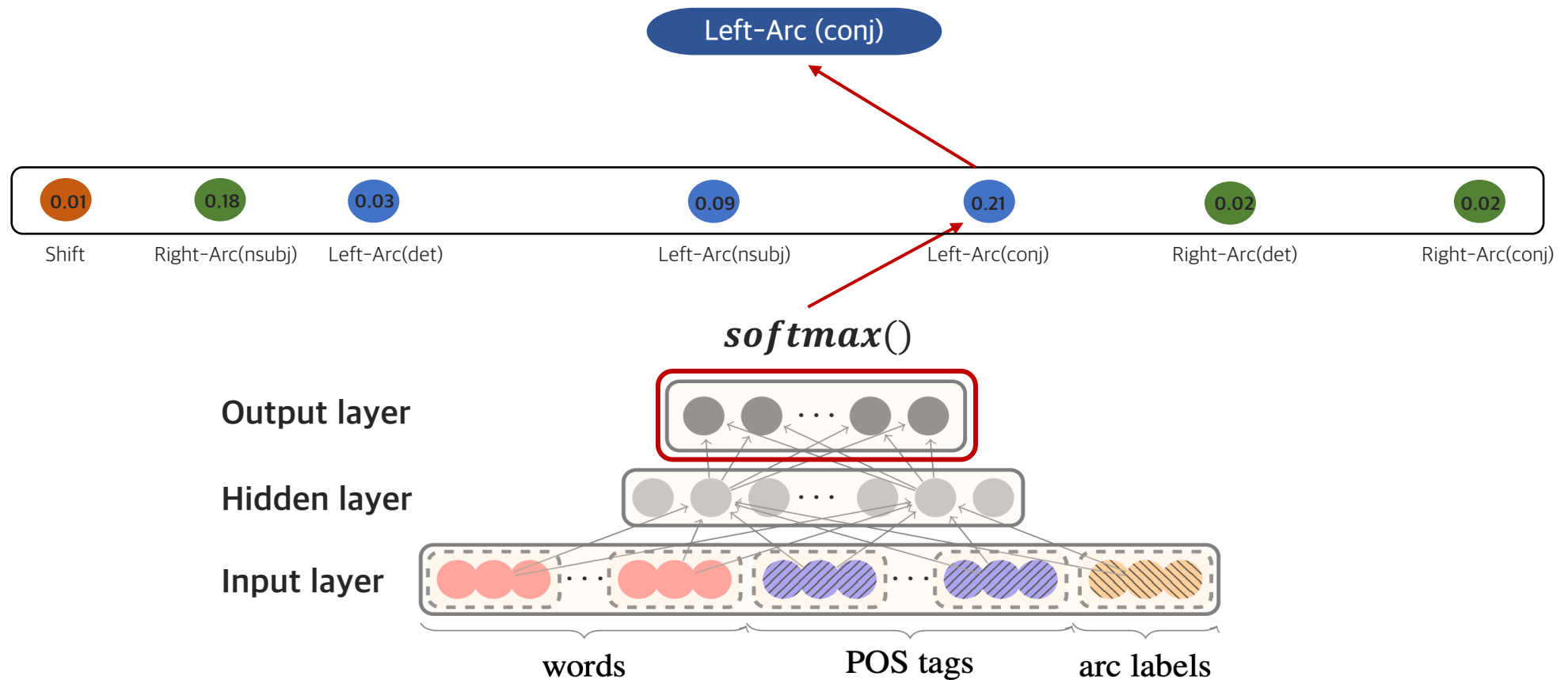
## Unit 04 | Neural dependency parsing

### Neural Dependency Parser - Output layer



## Unit 04 | Neural dependency parsing

### Neural Dependency Parser - Output layer



# Unit 04 | Neural dependency parsing

## Experiments - Results

### Evaluation Measures

- Unlabeled Attachment Score (UAS) : Arc 방향만 예측
- Labeled Attachment Score (LAS) : Arc 방향과 Label 예측

### Results

Parser	UAS	LAS	Sentence / sec	비고
MaltParser (2007)	89.8	87.2	469	Conventional Feature Representation
MSTParser (2007)	91.4	88.1	10	Graph-based
TurboParser (2010)	92.3	89.6	8	Graph-based
Neural Network Based Parser (2014)	92.0	89.7	654	Neural Network

# Unit 04 | Neural dependency parsing

## Experiments - Results

### Evaluation Measures

- Unlabeled Attachment Score (UAS) : Arc 방향만 예측
- Labeled Attachment Score (LAS) : Arc 방향과 Label 예측

### Results

Parser	UAS	LAS	Sentence / sec	비고
MaltParser (2007)	89.8	87.2	469	Conventional Feature Representation
MSTParser (2007)	91.4	88.1	10	Graph-based
TurboParser (2010)	92.3	89.6	8	Graph-based
Neural Network Based Parser (2014)	92.0	89.7	654	Neural Network-based
Weiss et al. (2015)	92.0	89.7		Greedy transition-based parser
Andor et al. 2016	94.61	92.79		Beam search
Dozat & Manning 2017	95.74	94.08		Neural graph-based

## 참고자료

- KoreaUniv DSBA, [\[DSBA\]CS224N-05.Linguistic Structure: Dependency Parsing](#), 노영빈
- dongtta's Blog, [CS224n] 5강 - Dependency Parsing
- 제이의 블로그, [\[cs224n\]Lecture 6. Dependency parsing](#)
- [A Fast and Accurate Dependency Parser using Neural Networks](#), Chen & Manning, 2014



**Q & A**

들어주셔서 감사합니다.