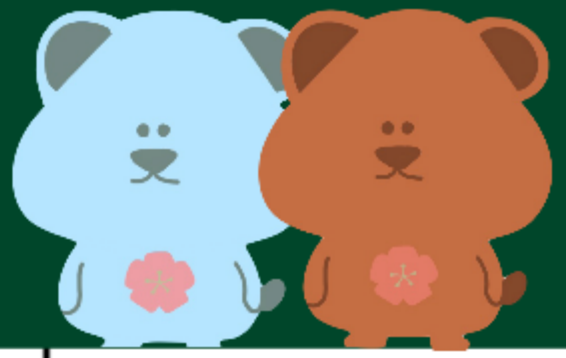


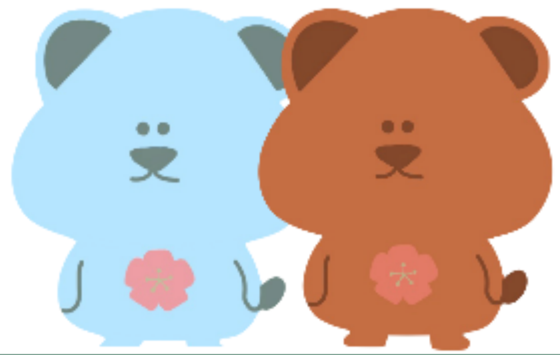
Week 05.
Linguistic Structure - Dependency parsing

발표자: 김소민, 임세영

EWHA EURON

목차

Parsing



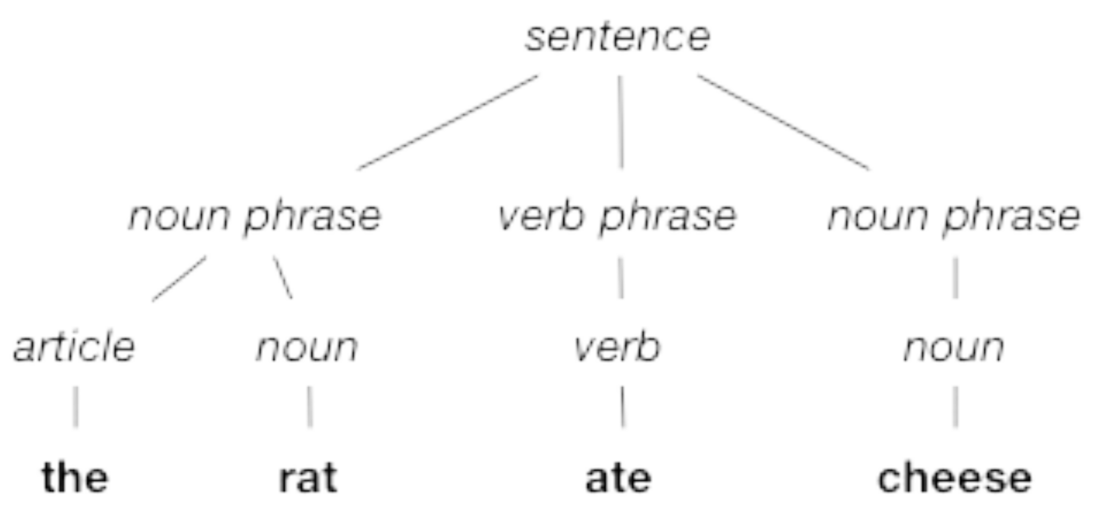
Parsing 정의

Parsing 이란?

각 문장의 문법적인 구성 또는 구문을 분석하는 과정 -> 구문분석 트리를 구성하는 것

- Constituency Parsing
  문장 구조 파악하기

- Dependency Parsing
  단어 간 관계를 파악하기



Tokenizing vs Pos Tagging vs Parsing

Tokenizing

텍스트에 대해 특정 기준 단위로
문장을 나누는 것을 의미

Ex)
- 문장을 단어로
- 글을 문장 단위로

Pos Tagging

품사 태깅(Part-of-Speech Tagging)

토큰(Token)들에게 품사를 붙여주는
작업을 뜻한다.

Tokenizing vs Pos Tagging vs Parsing

Tokenizing

```
test_text = ['All', 'rights', 'reserved', '.']
```

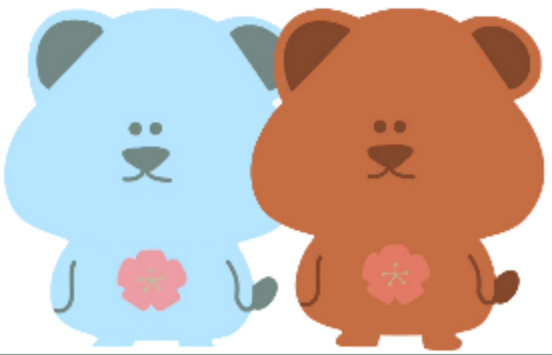
Pos Tagging

```
[[('All', 'DT'), ('rights', 'NNS'), ('reserved', 'VBN'),
('.', '.')]]
```

기호	의미	
CC	등위 접속사	coordinating conjunction
CD	기본 숫자	cardinal digit
DT	한정사	determiner
EX	existential there (장소를 뜻하는 there이 아닌, '~이 있다'라는 뜻의 there)	existential there (like: "there is" ... think of it like "there exists")
FW	외래어	foreign word
IN	전치사/종속 접속사	preposition/subordinating conjunction
JJ	형용사 // 'big'	adjective 'big'
JJR	비교 형용사 // 'bigger'	adjective, comparative 'bigger'
JJS	최상급 형용사 // 'biggest'	adjective, superlative 'biggest'
LS	리스트 마커 // 1)	list marker 1)
MD	조동사 // could, will	modal could, will
NN	단수 명사 // 'desk'	noun, singular 'desk'
NNS	복수 명사 // 'desks'	noun plural 'desks'

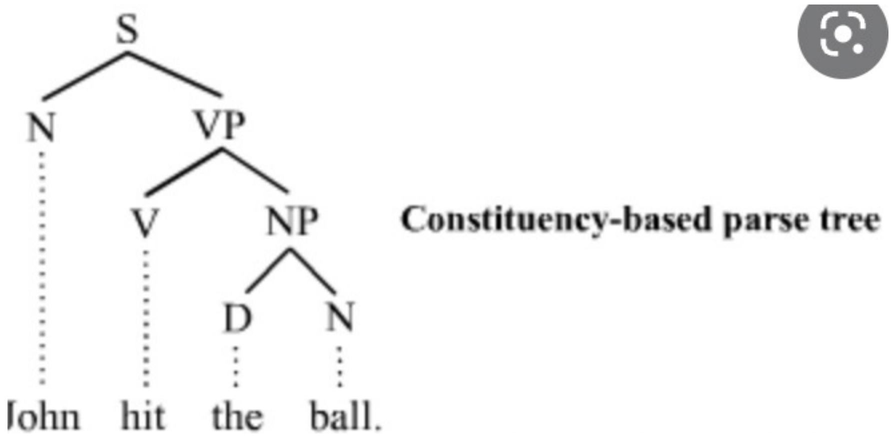
EWHA EURON

Constituency Parsing

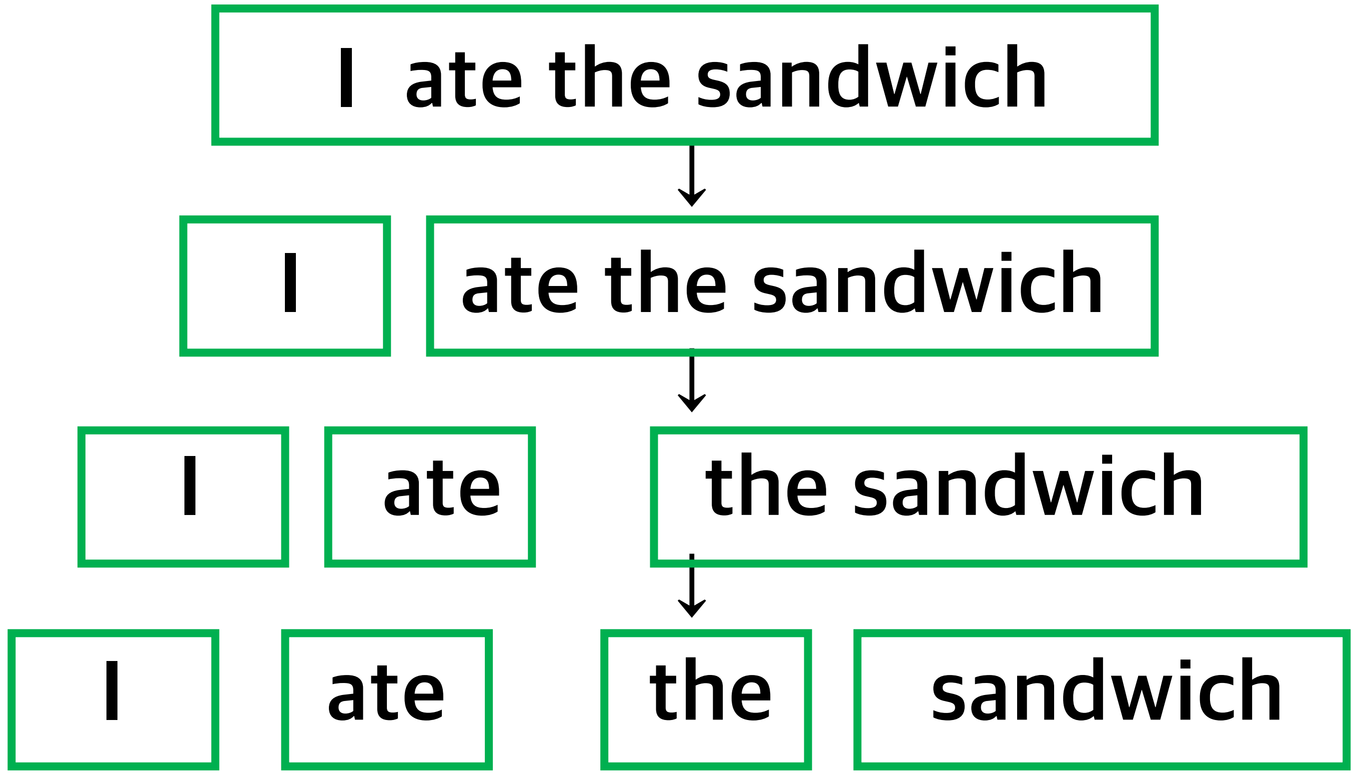


Constituency Parsing

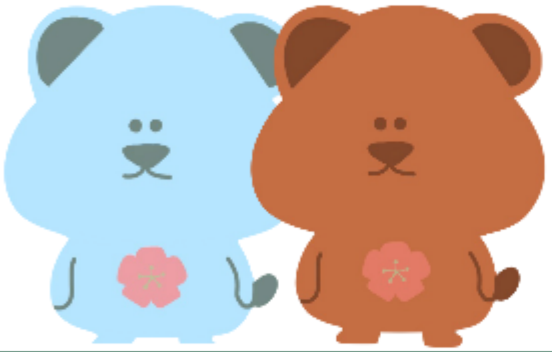
- 문장을 중첩된 성분으로 나누어 문장의 구조를 파악
- 어순이 고정적인 언어에서 주로 사용 (ex. 영어)
- 재귀적으로 적용 가능
- css224n 18강에서 추후 다뤄짐



S
N VP
V NP
D N



Dependency Parsing



Dependency Parsing

- 단어 간 관계 파악
  -> 각 단어 간 의존 or 수식 관계를 파악 가능
- 자유 어순을 가지거나 문장 성분이 생략 가능한 언어에서 주로 사용됨 (ex. 한국어)

I ate the sandwich.

- 화살표가 향하는 방향: 수식 하는 방향

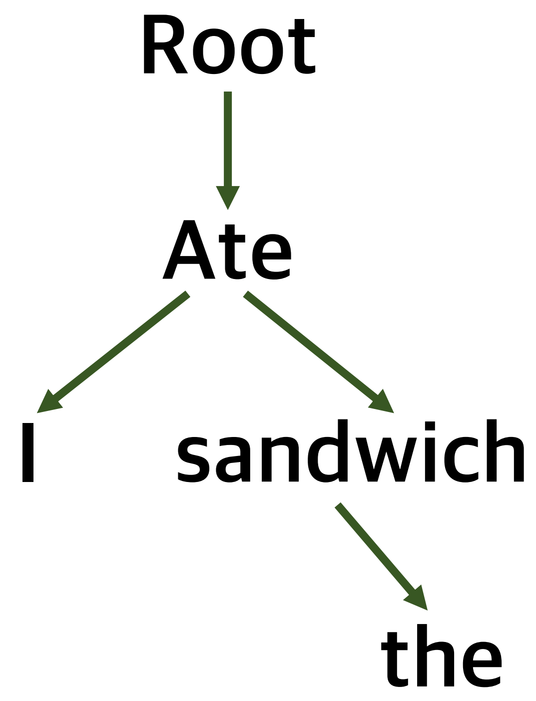
ate	-----→	I	(Subject)
ate	-----→	sandwich	(Object)
sandwich	-----→	the	(determiner)

Head	Dependent
governor	Modifier

Dependency Parsing

- 단어 간 관계 파악
  -> 각 단어 간 의존 or 수식 관계를 파악 가능
- 자유 어순을 가지거나 문장 성분이 생략 가능한 언어에서 주로 사용됨 (ex. 한국어)

I ate the sandwich.



Dependency 파싱이 필요한 이유

언어를 올바르게 이해하기 위해서

→ 인간들은 작은 단어들을 큰 단어로 조합함으로써
복잡한 아이디어를 표현하고 전달함

→ 어떤 단어가 무엇과 연결되었는지 정확히 이해해야,
그 의미를 정확하게 알 수 있음

파싱이 필요한 이유 - ambiguity

Coordination Scope Ambiguity

- 특정 단어가 수식하는 대상의 범위가 모호
- 중의적으로 해석 *작용역 중의성

[Shuttle veteran and longtime Nasa executive] Fred Gregory appointed to board.
우주선 베테랑이자 오랜 나사의 임원인 Fred Gregory가 이사로 임명 되었다.

[Shuttle veteran] and [longtime Nasa executive] Fred Gregory appointed to board.
우주선 베테랑과 오랜 나사의 임원인 Fred Gregory가 이사로 임명 되었다.

파싱이 필요한 이유 - ambiguity

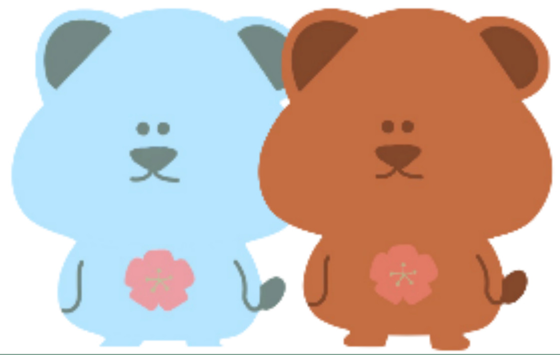
Phrase Attachment Ambiguity

- 형용사구, 동사구, 전치사구 등이 어떤 단어를 수식하는지에 따라 의미가 모호함

San Jose cops kill man with knife

1. 산호세 경찰들은 칼을 든 남자를 죽였다.
2. 산호세 경찰들은 남자를 칼로 죽였다.

04 Dependency Grammar

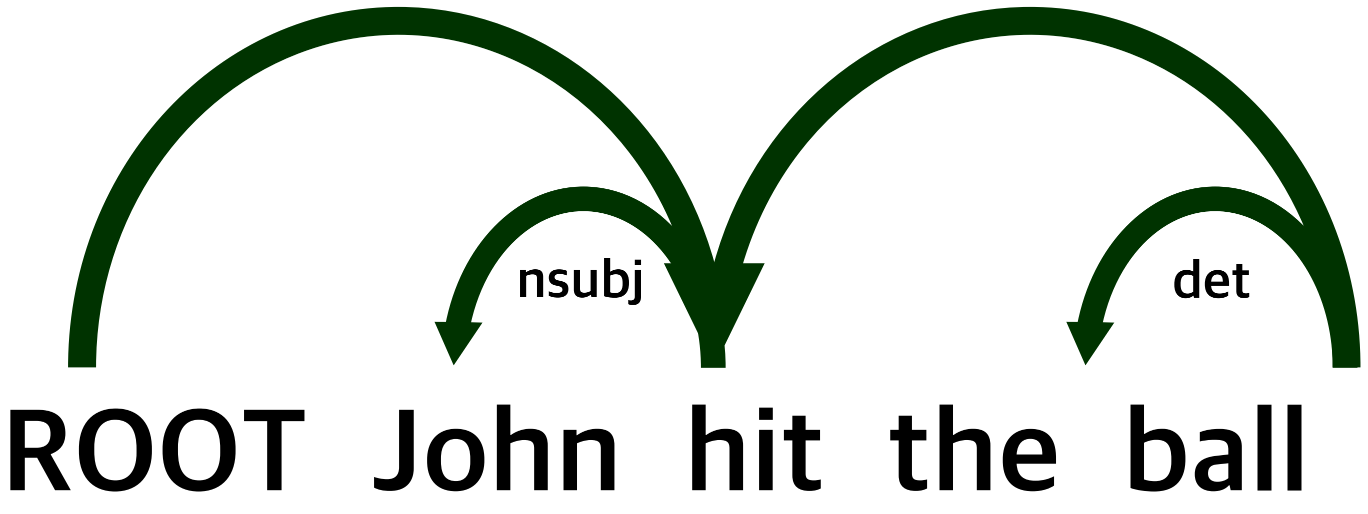


EWHA EURON

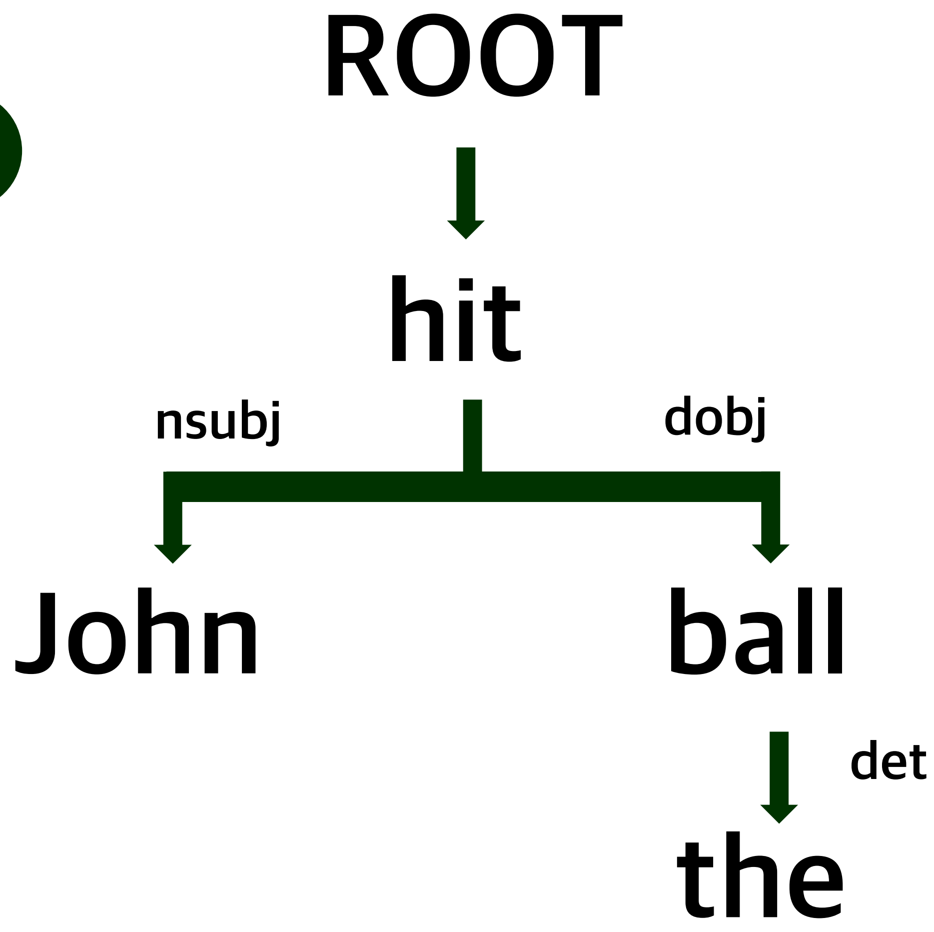
04 Dependency Grammar

#1 Grammar and Structure

1



2



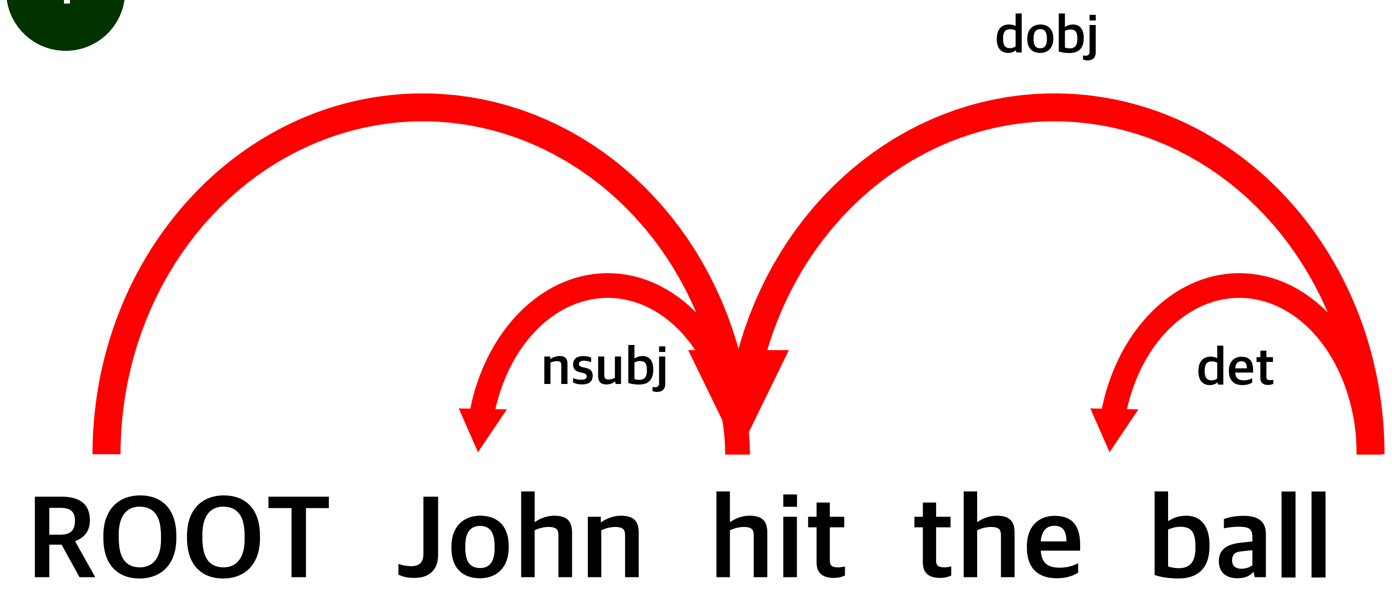
Dependency Structure는 두가지 형태로 표현 가능

1. 수식하는 단어를 화살표로 표현하는 방식
2. 트리 형태로 parsing output을 도출하는 방식

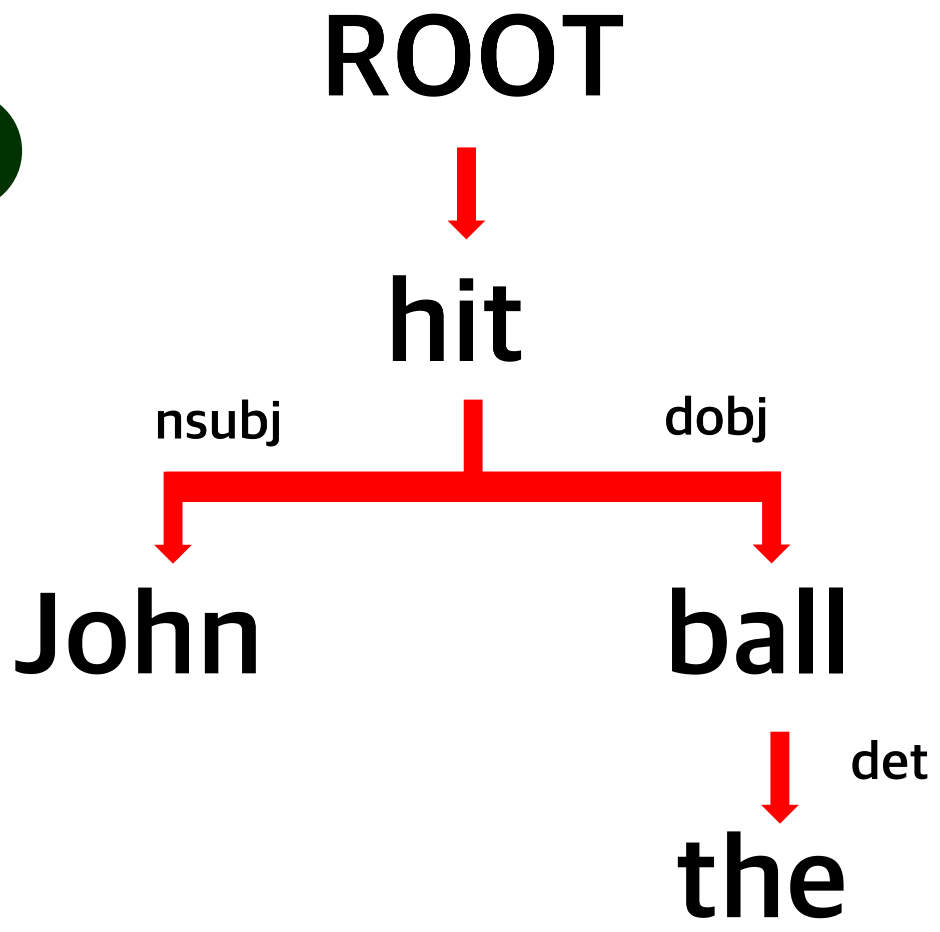
04 Dependency Grammar

#1 Grammar and Structure

1



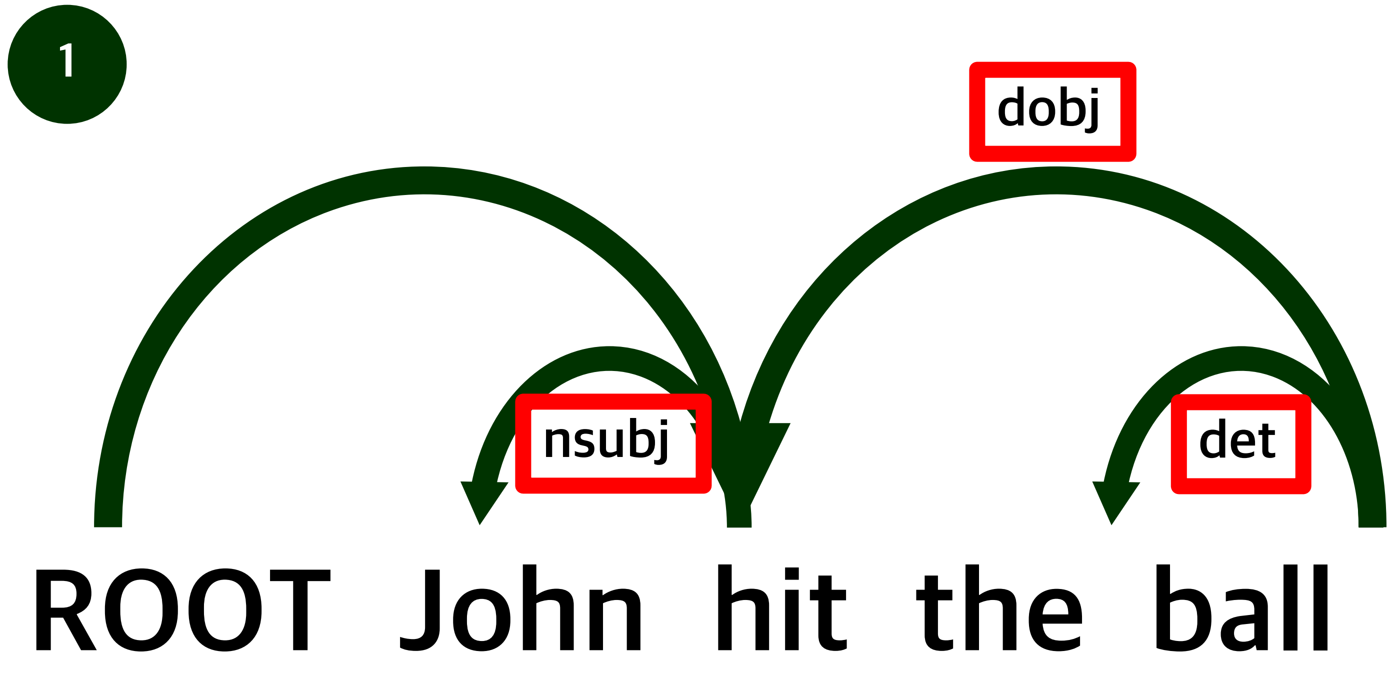
2

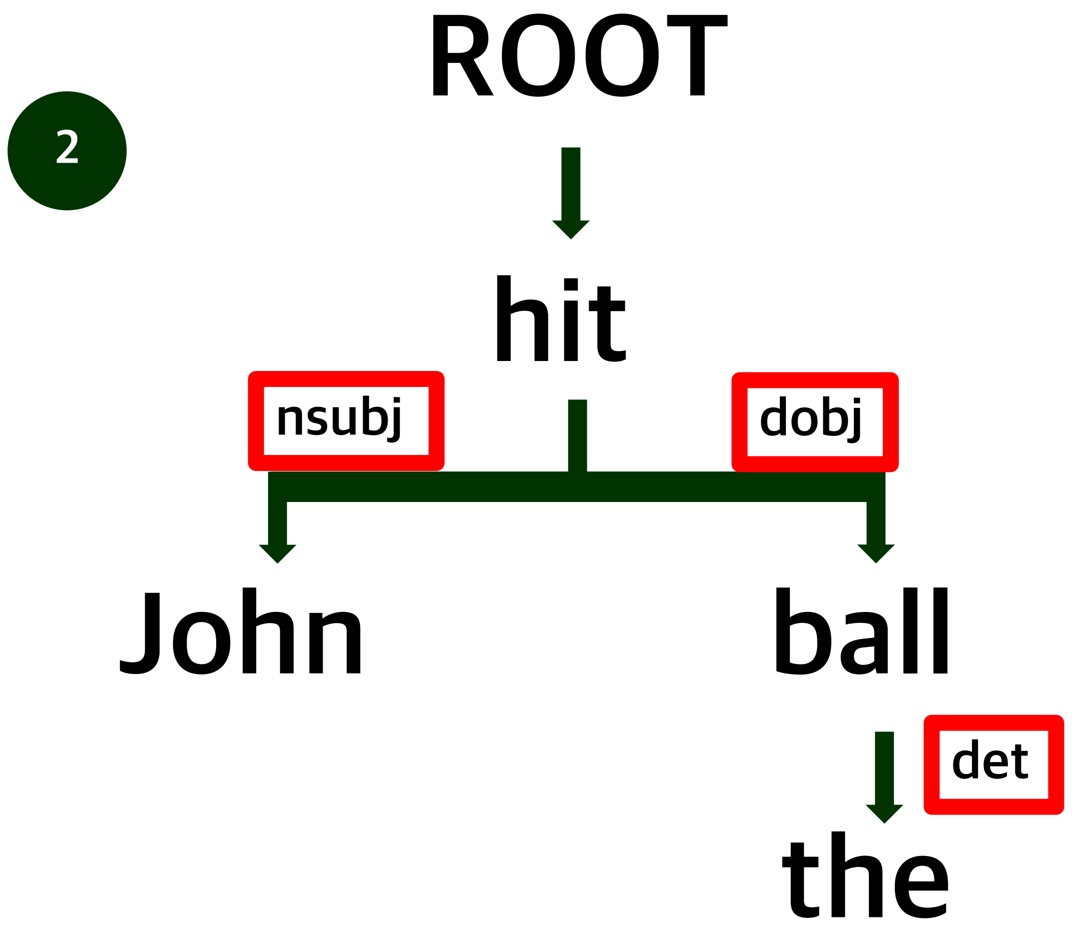


화살표: head > dependent
(수식을 받는 단어) (수식을 하는 단어)

04 Dependency Grammar

#1 Grammar and Structure



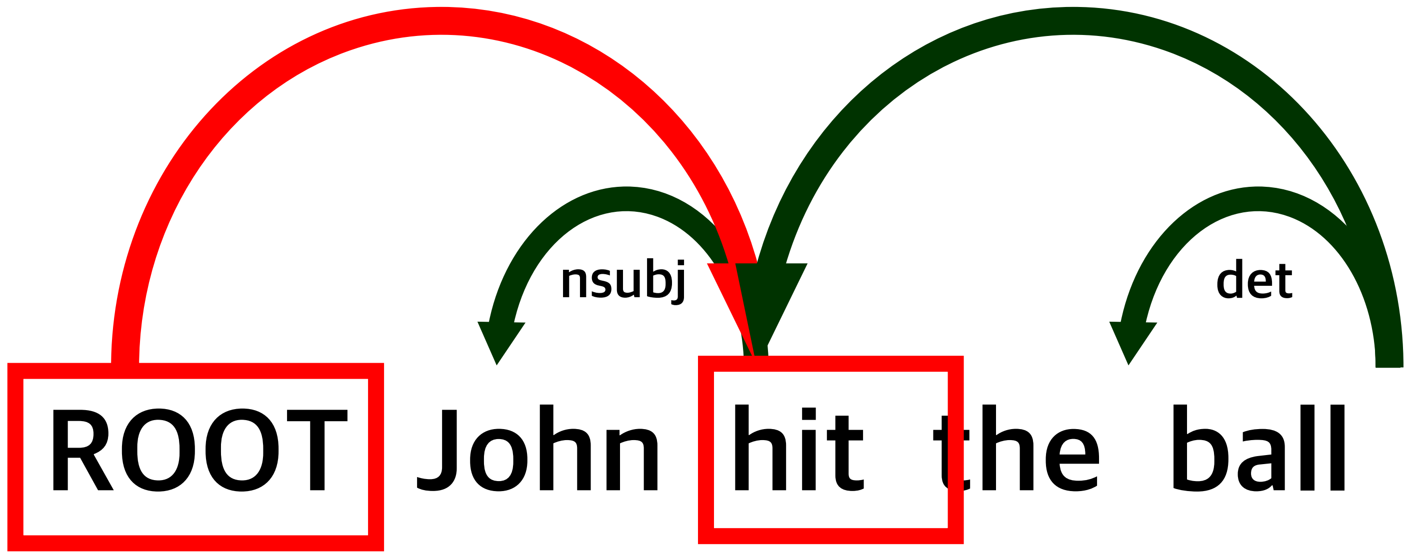


화살표 위 label은 dependency (단어간 문법적 관계)를 의미

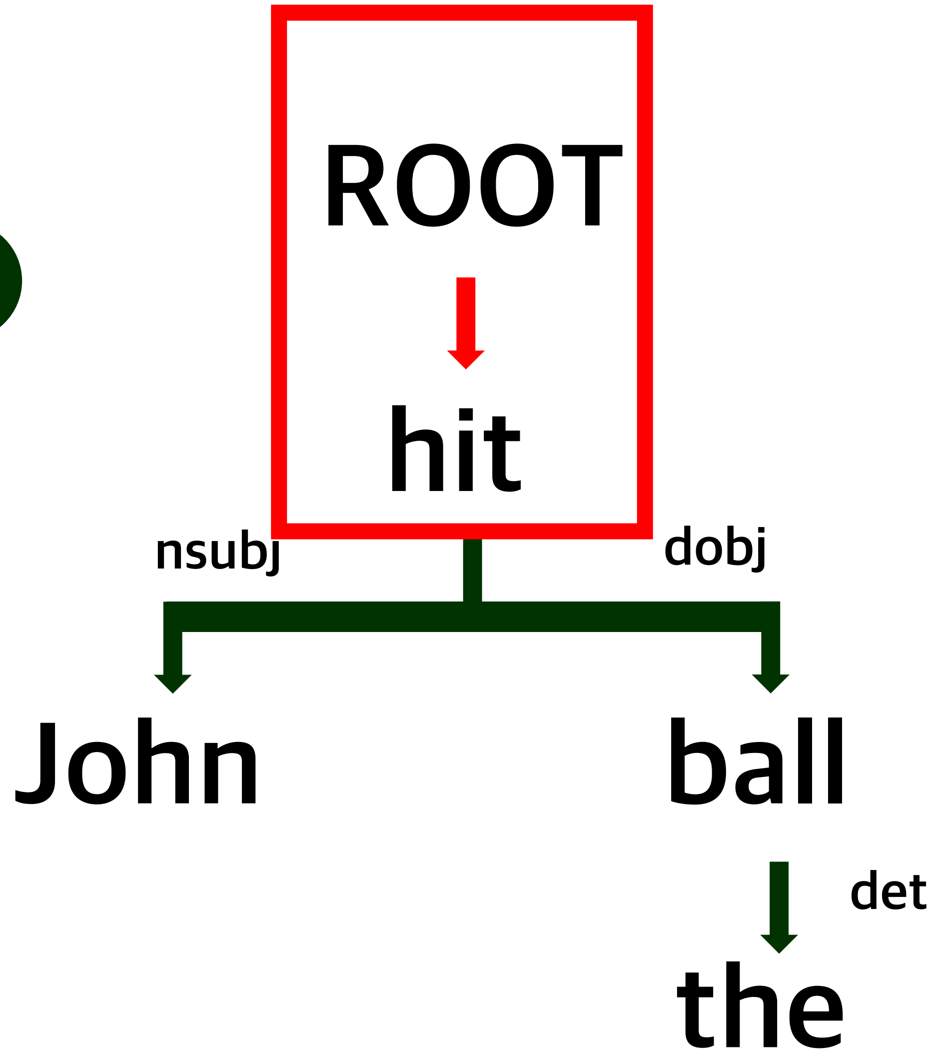
04 Dependency Grammar

#1 Grammar and Structure

1



2



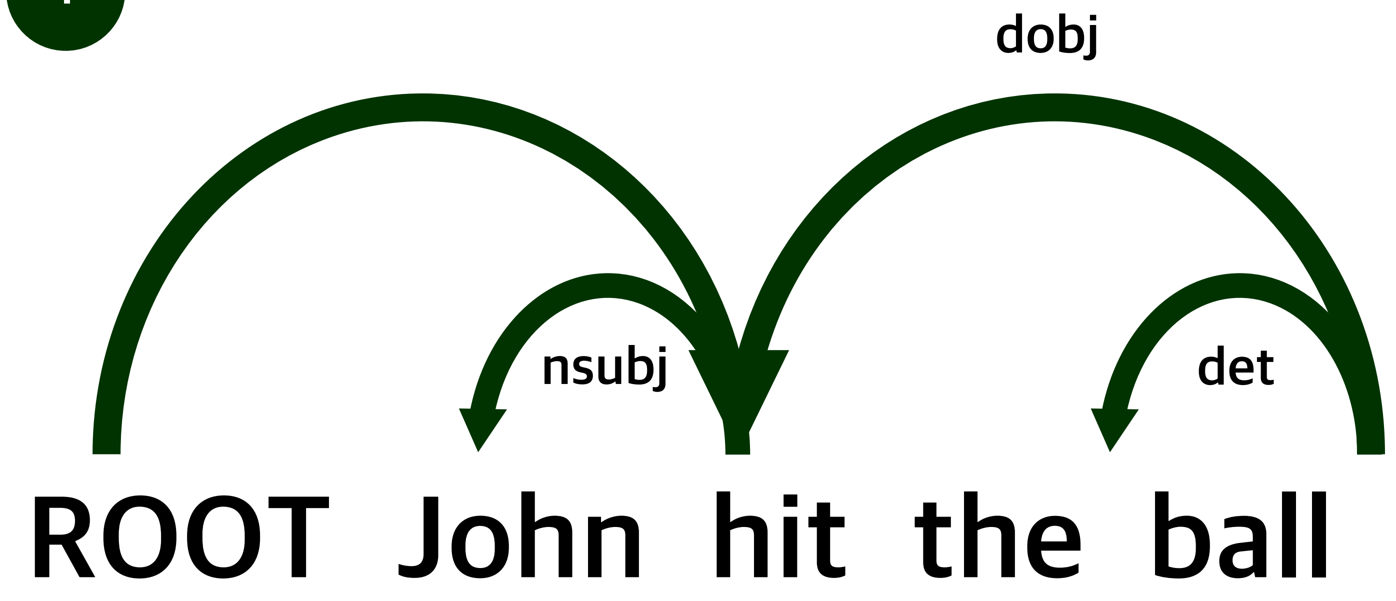
ROOT: 어떠한 단어의 수식도 받지 않는 단어를 위해 만들어짐

> 모든 단어가 최소한 1개 node의 dependent

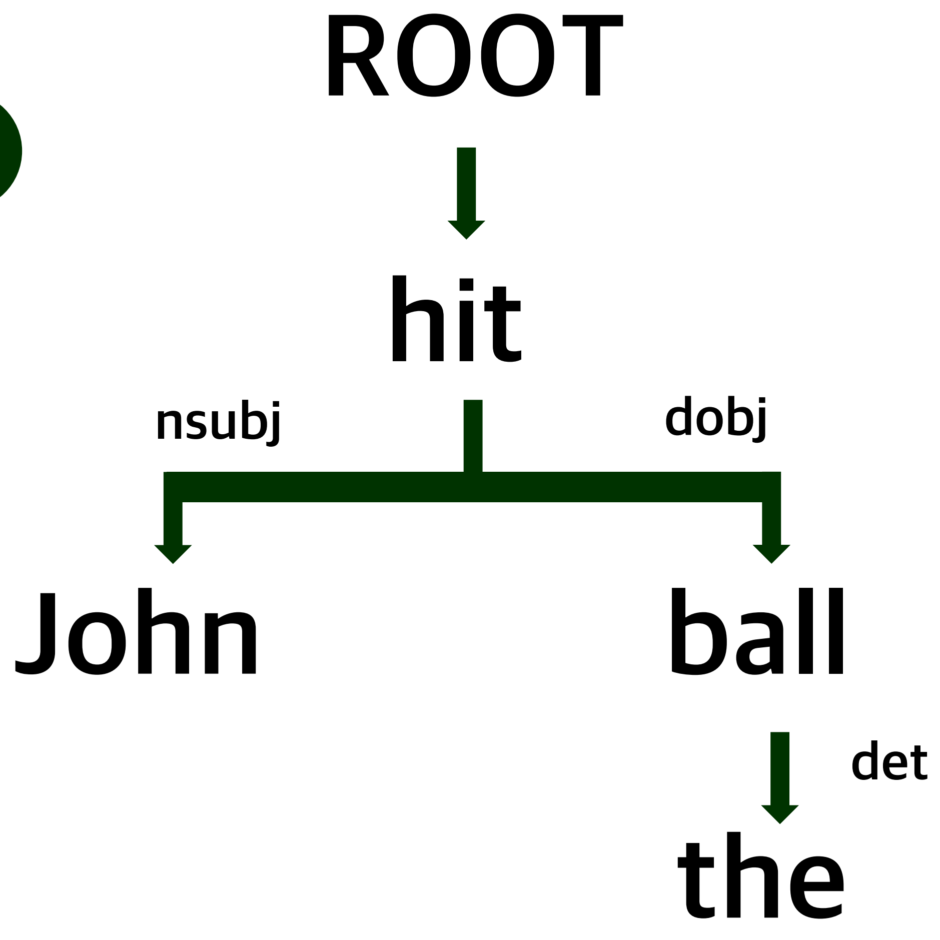
04 Dependency Grammar

#1 Grammar and Structure

1



2



화살표는 순환하지 않음
> Parsing 결과물을 tree 형태로 표현 가능

04 Dependency Grammar

#2 Universal Feature

What are the sources of information for dependency parsing?

1. Bilexical affinities
   - [issues > the] is plausible
2. Dependency distance
   - mostly with nearby words
3. Intervening material
   - Dependencies rarely span intervening verbs or punctuation
4. Valency of heads
   - How many dependents on which side are usual for a head?

04 Dependency Grammar

#3 Tree Bank Dataset

Figure 1: Slider for selecting sentiments in range 0-25

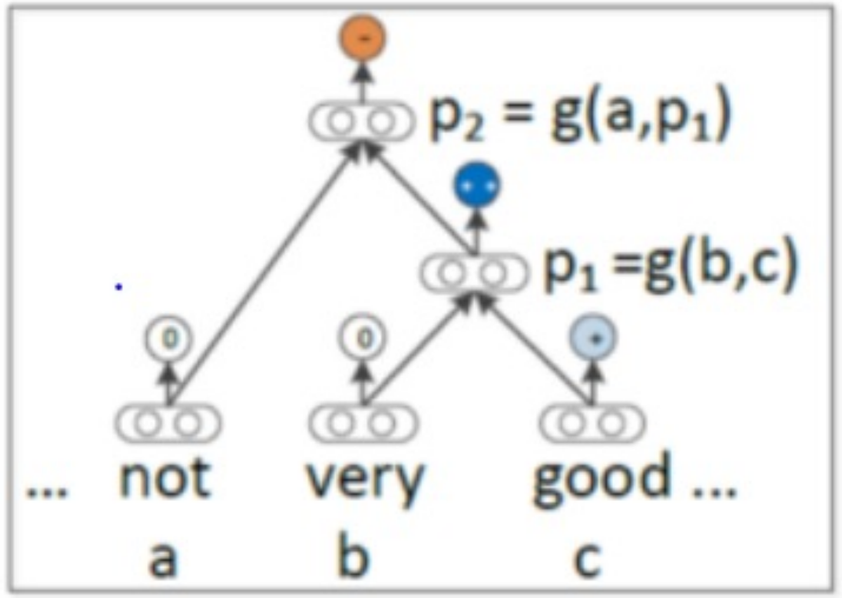


Figure 2: Recursive models computing parent vector in a bottom up approach

짧은 문장의 극성을 예측하고 문장의 어순을 무시하는 bag-of-words 접근 방식
> 어려운 부정 예제를 분류하는 효율적인 모델 생성

감성 분석 작업의 이진 분류 정확도 상승

05 Dependency Parsing Method

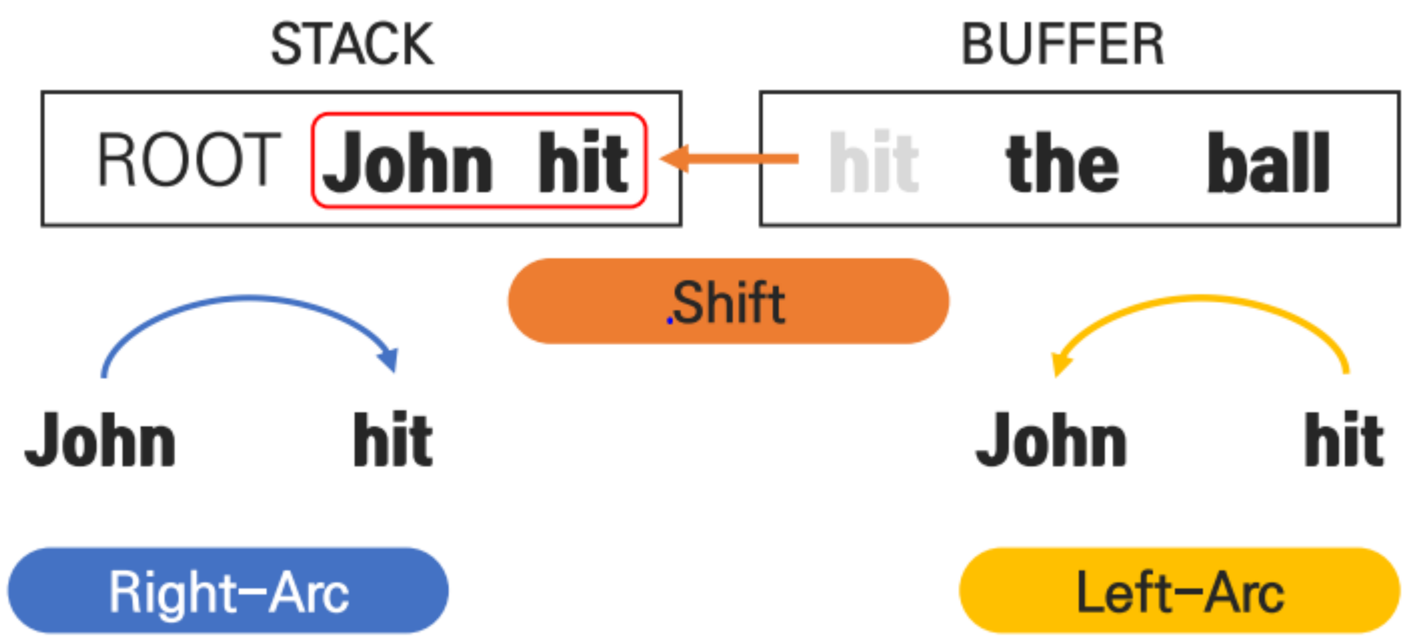
05 Dependency Parsing Method

#1 Main Method

1

Transition-based

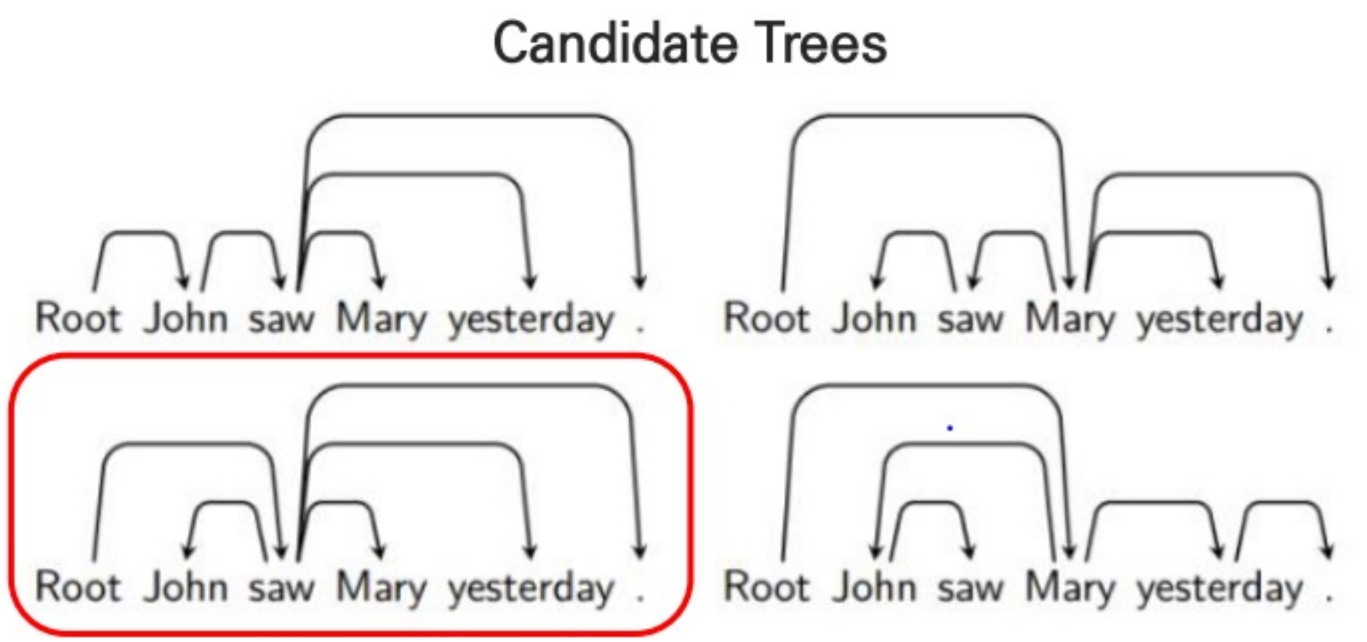
두 단어의 의존여부를 순서대로 결정하며
점진적으로 구문분석 트리를 구성



2

Graph-based

가능한 의존 관계를 모두 고려한 뒤
가장 확률이 높은 구문분석 트리 선택



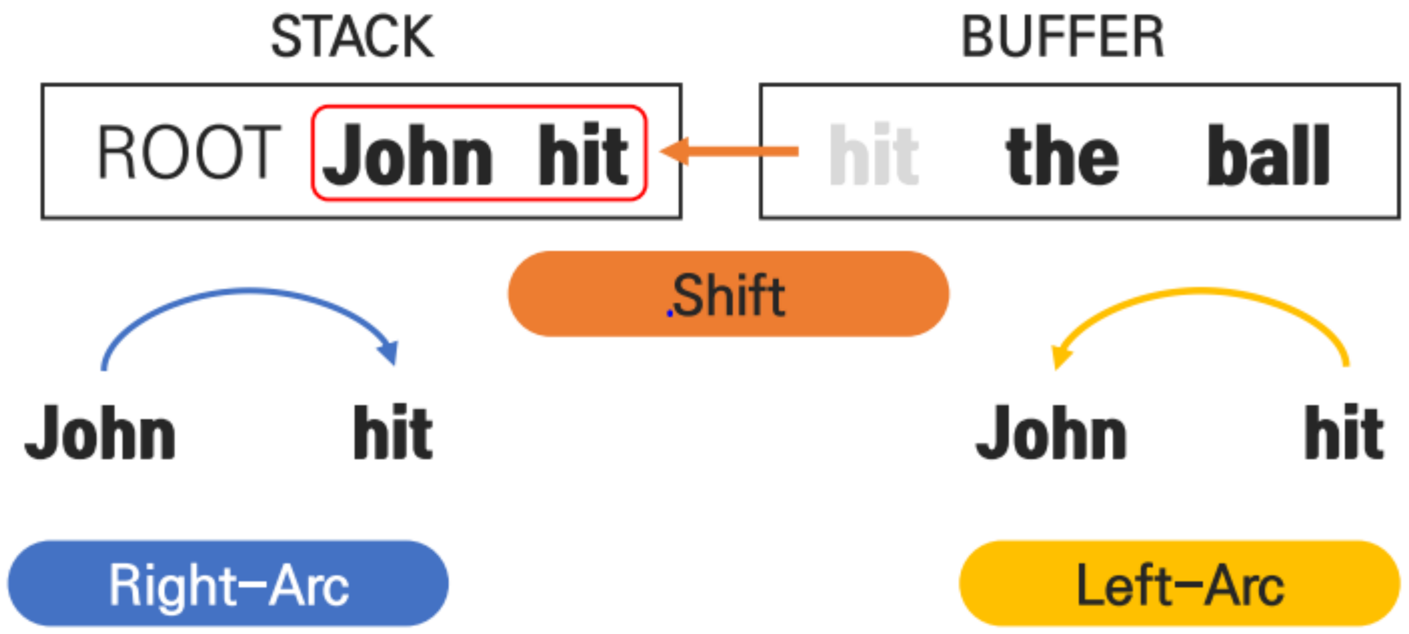
05 Dependency Parsing Method

#1 Main Method

1

Transition-based

두 단어의 의존여부를 순서대로 결정하며
점진적으로 구문분석 트리를 구성



Graph-based에 비해

속도가 빠르지만 정확도가 낮음

05 Dependency Parsing Method

#2 Transition-based Parsing

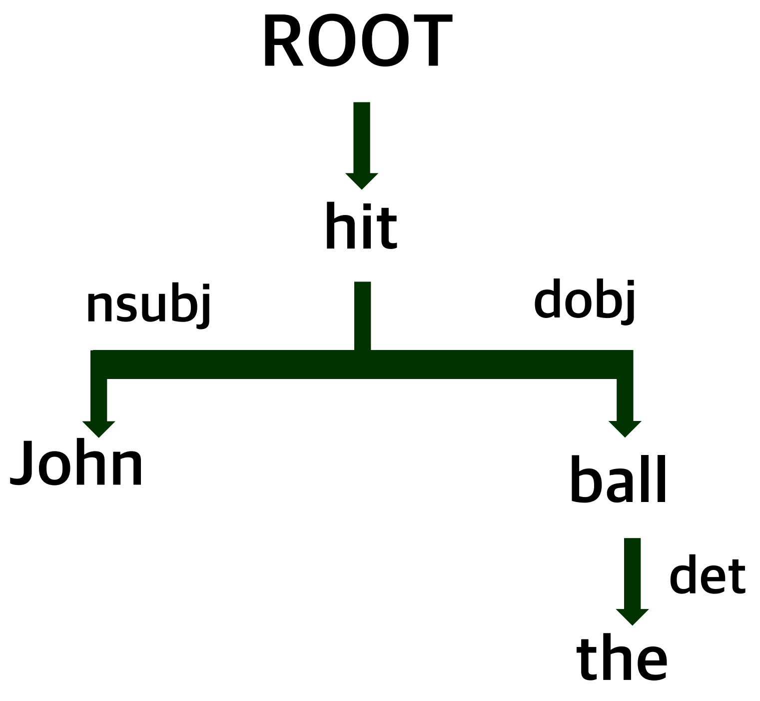
Input Sentence

"John hit the ball"

$c = (\sigma, \beta, A)$

State (c)

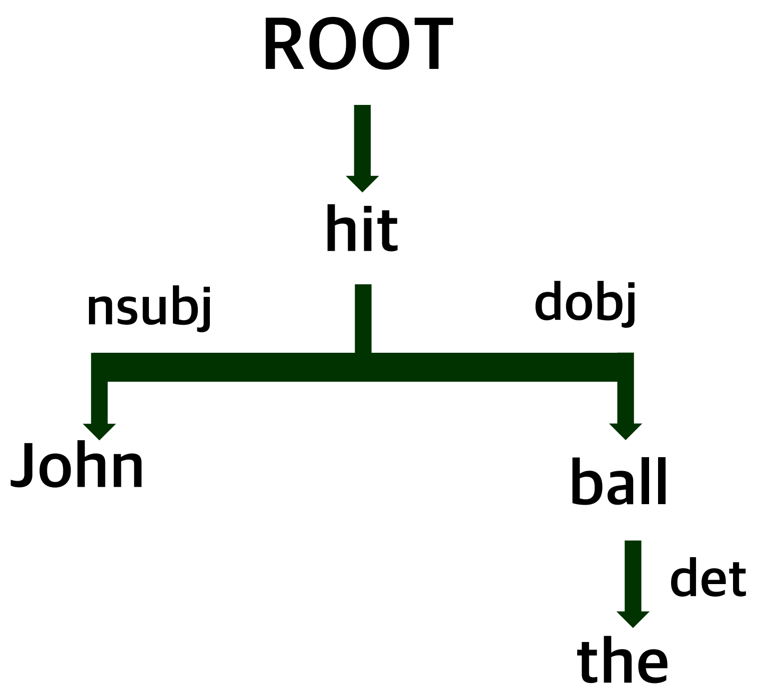
모든 decision은 State c 를 input으로 하는
함수 f(c) 를 통해 이루어짐 (ex: SVM, NN)



STACK (σ)	BUFFER (β)	Set of Arcs (A)
ROOT	John hit the ball	∅

05 Dependency Parsing Method

#2 Transition-based Parsing



Input Sentence

"John hit the ball"

$c = (\sigma, \beta, A)$

State (c)

decision

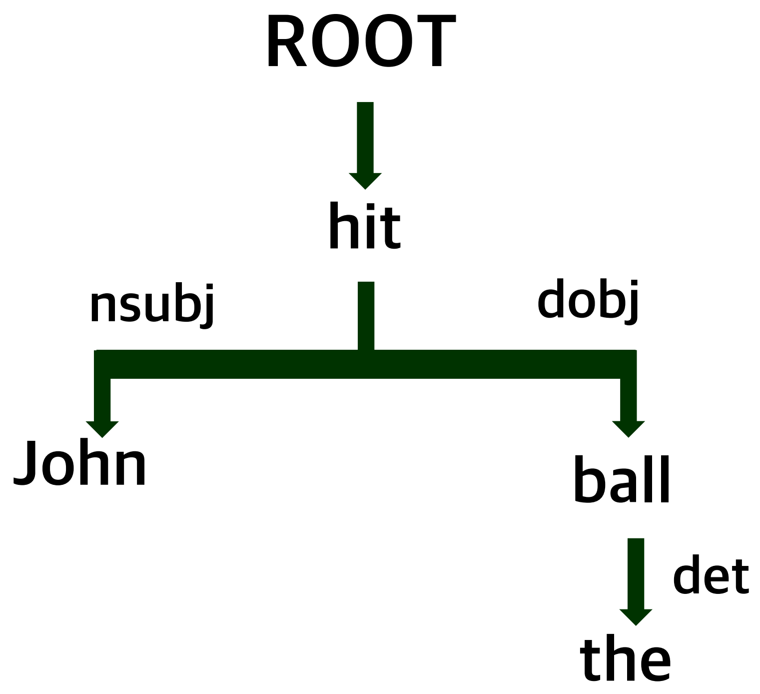
1. Shift — 1 개
2. Right-Arc nsubj (주어) Dobj (직접목적어) det (한정사) acomp (형용사보어) — N_l 개
3. Left-Arc nsubj (주어) Dobj (직접목적어) det (한정사) acomp (형용사보어) — N_l 개

> $2N_l + 1$ 개

ROOT	John hit the ball	∅
STACK (σ)	BUFFER (β)	Set of Arcs (A)

05 Dependency Parsing Method

#2 Transition-based Parsing



Right-arc 또는 Left-arc Decision은 Stack top 1,2 단어를 대상으로 이루어짐

Input Sentence
"John hit the ball"

$c = (\sigma, \beta, A)$

State (c)

STACK (σ)	BUFFER (β)	Set of Arcs (A)
ROOT John	John hit the ball	∅

Decision Process

shift

05 Dependency Parsing Method

#2 Transition-based Parsing

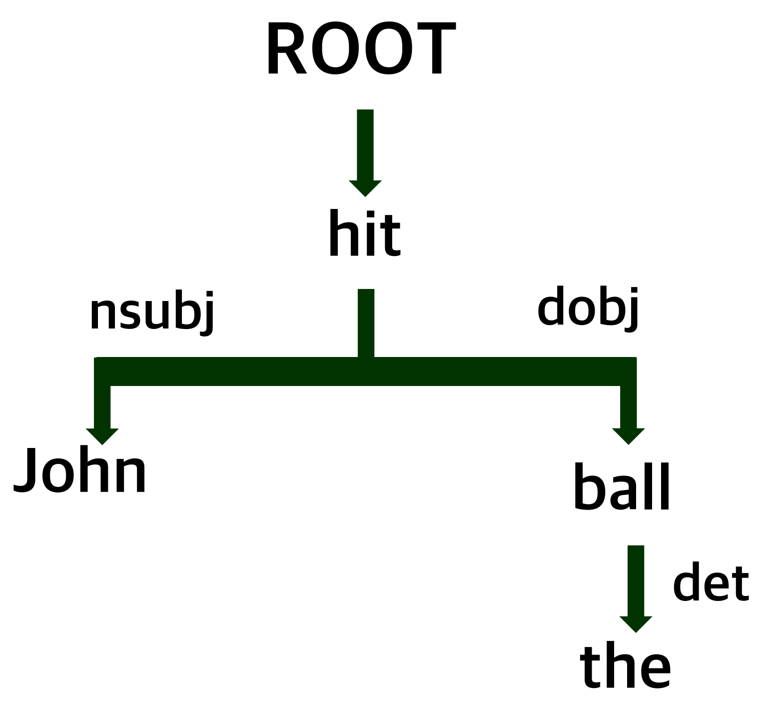
ROOT token은 BUFFER가 비워질 때까지 decision의 대상이 되지 않음

Input Sentence

"John hit the ball"

$c = (\sigma, \beta, A)$

State (c)



ROOT John hit

John hit the ball

∅

STACK (σ)

BUFFER (β)

Set of Arcs (A)

Decision Process

shift -> shift

05 Dependency Parsing Method

#2 Transition-based Parsing

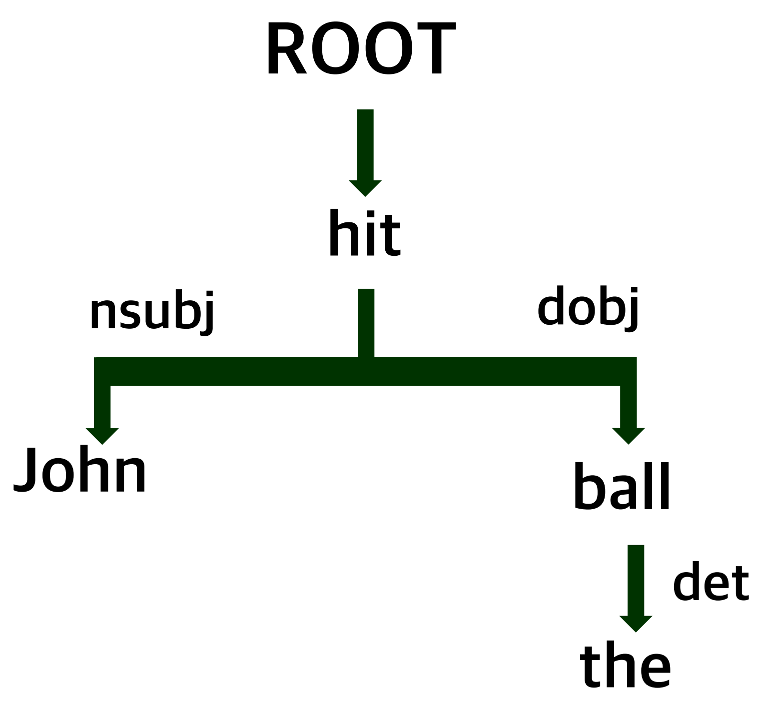
Input Sentence

"John hit the ball"

$c = (\sigma, \beta, A)$

State (c)

f(c)



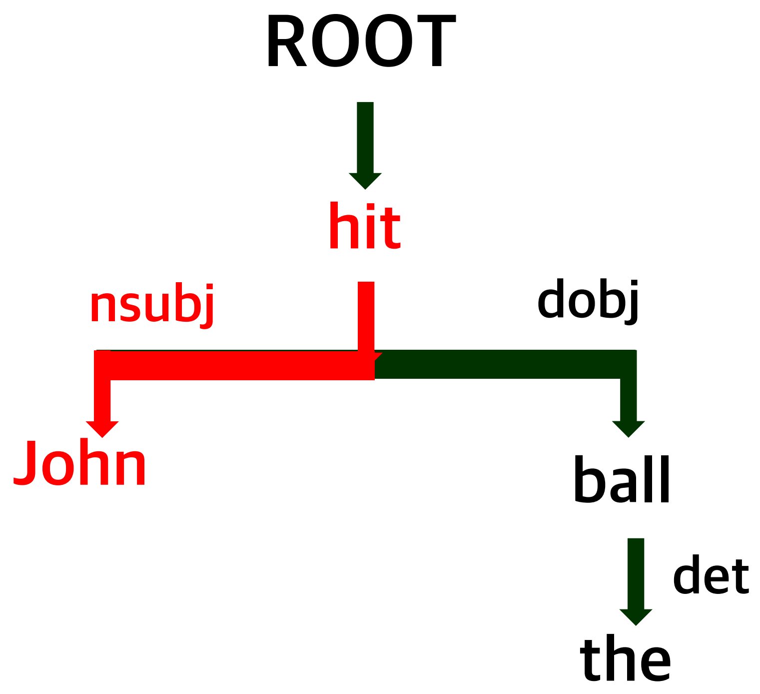
ROOT (John hit)	John hit the ball	∅
STACK (σ)	BUFFER (β)	Set of Arcs (A)

Decision Process

shift -> shift -> ?

05 Dependency Parsing Method

#2 Transition-based Parsing



Input Sentence

“John hit the ball”

$c = (\sigma, \beta, A)$

State (c)

remove

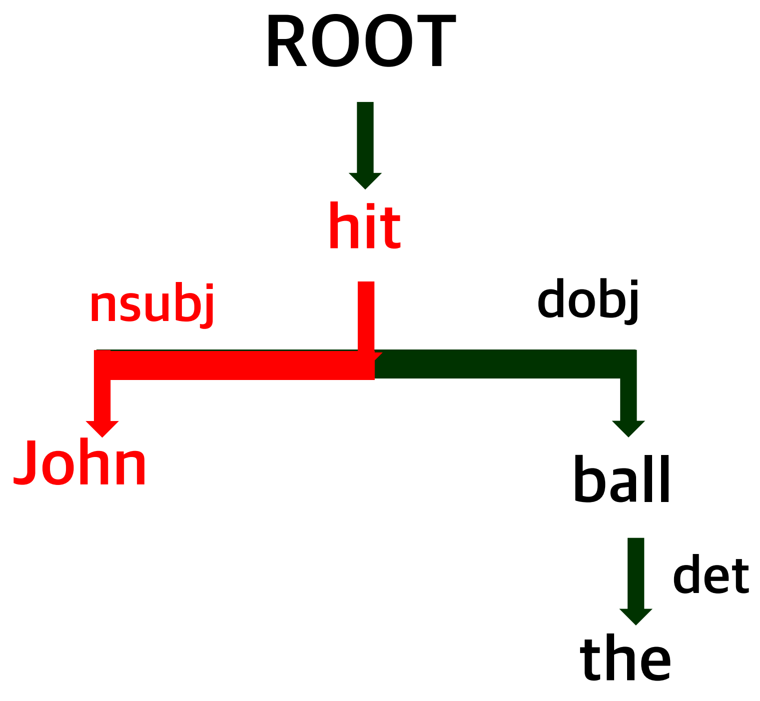
STACK (σ)	BUFFER (β)	Set of Arcs (A)
ROOT John hit	John hit the ball	(hit, nnsubj, John)

Decision Process

shift -> shift -> Left-Arc (nsubj)

05 Dependency Parsing Method

#2 Transition-based Parsing



Input Sentence

"John hit the ball"

$c = (\sigma, \beta, A)$

State (c)

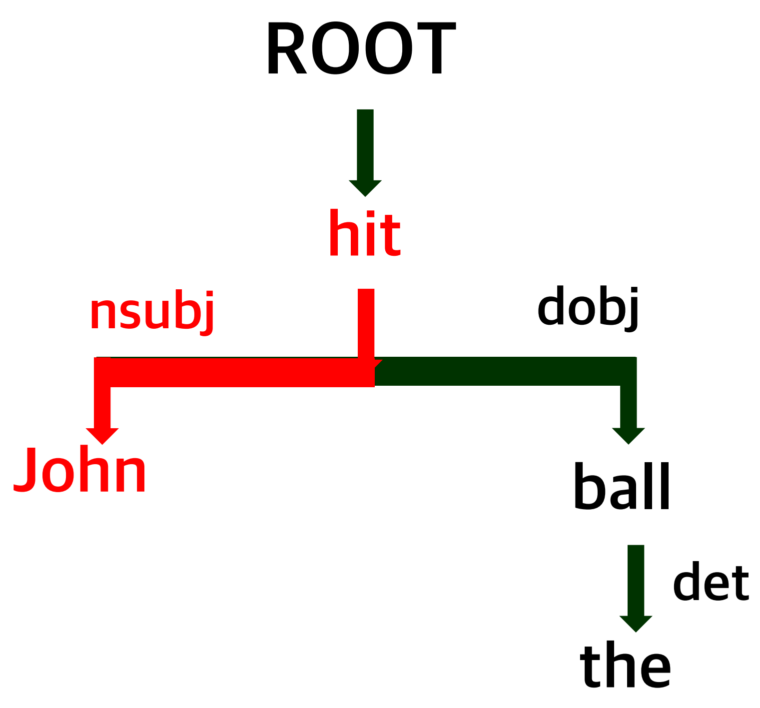
STACK (σ)	BUFFER (β)	Set of Arcs (A)
ROOT hit the	John hit the ball	(hit, nnsubj, John)

Decision Process

shift -> shift -> Left-Arc (nsubj) -> shift

05 Dependency Parsing Method

#2 Transition-based Parsing



Input Sentence

"John hit the ball"

$c = (\sigma, \beta, A)$

f(c)

State (c)

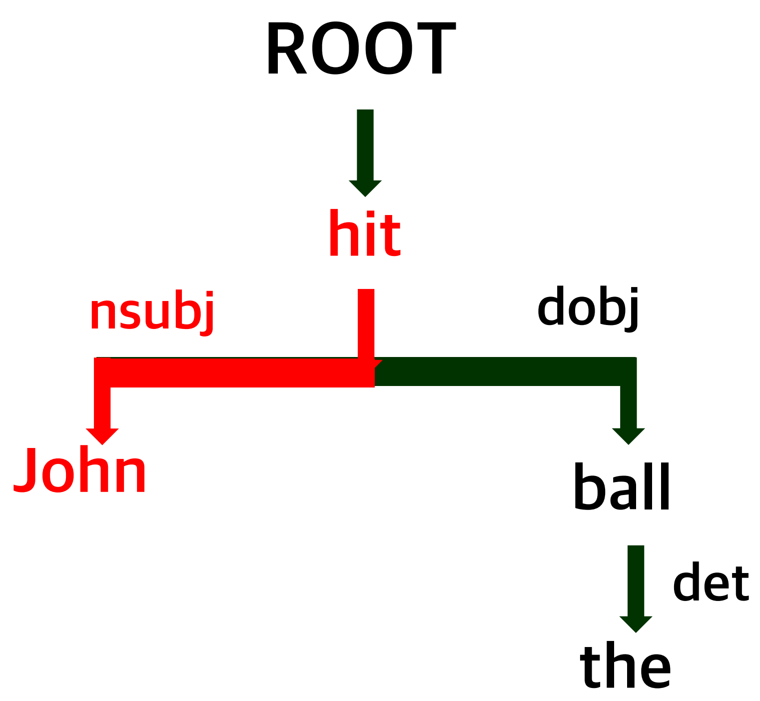
STACK (σ)	BUFFER (β)	Set of Arcs (A)
ROOT (hit the)	John hit the ball	(hit, nnsubj, John)

Decision Process

shift -> shift -> Left-Arc (nsubj) -> shift -> ?

05 Dependency Parsing Method

#2 Transition-based Parsing



Input Sentence

“John hit the ball”

$c = (\sigma, \beta, A)$

State (c)

STACK (σ)	BUFFER (β)	Set of Arcs (A)
ROOT hit the ball	John hit the ball	(hit, nnsubj, John)

Decision Process

shift -> shift -> Left-Arc (nsubj) -> shift -> shift

05 Dependency Parsing Method

#2 Transition-based Parsing

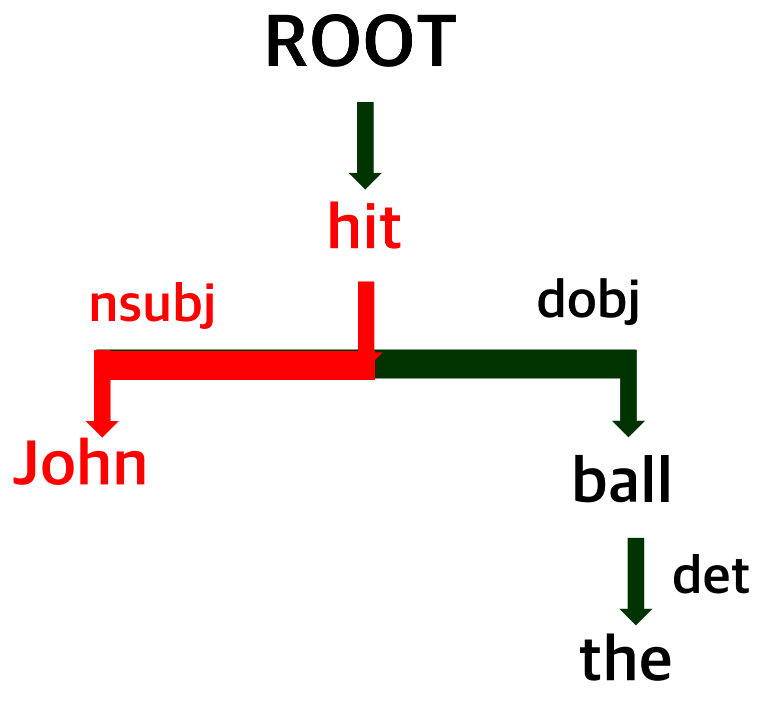
Input Sentence

"John hit the ball"

$c = (\sigma, \beta, A)$

State (c)

f(c)



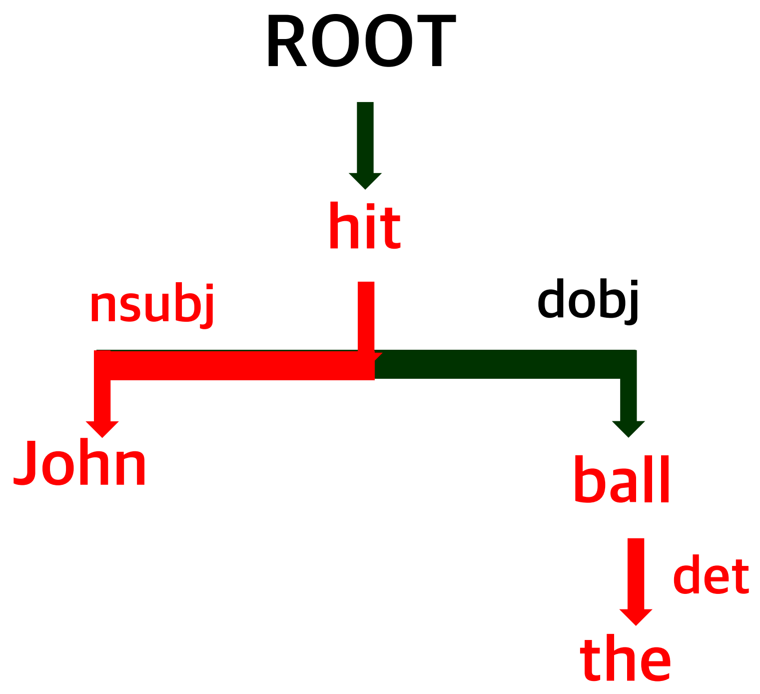
STACK (σ)	BUFFER (β)	Set of Arcs (A)
ROOT hit (the ball)	John hit the ball	(hit, nnsubj, John)

Decision Process

shift -> shift -> Left-Arc (nsubj) -> shift -> shift -> ?

05 Dependency Parsing Method

#2 Transition-based Parsing



Input Sentence

"John hit the ball"

$c = (\sigma, \beta, A)$

State (c)

remove

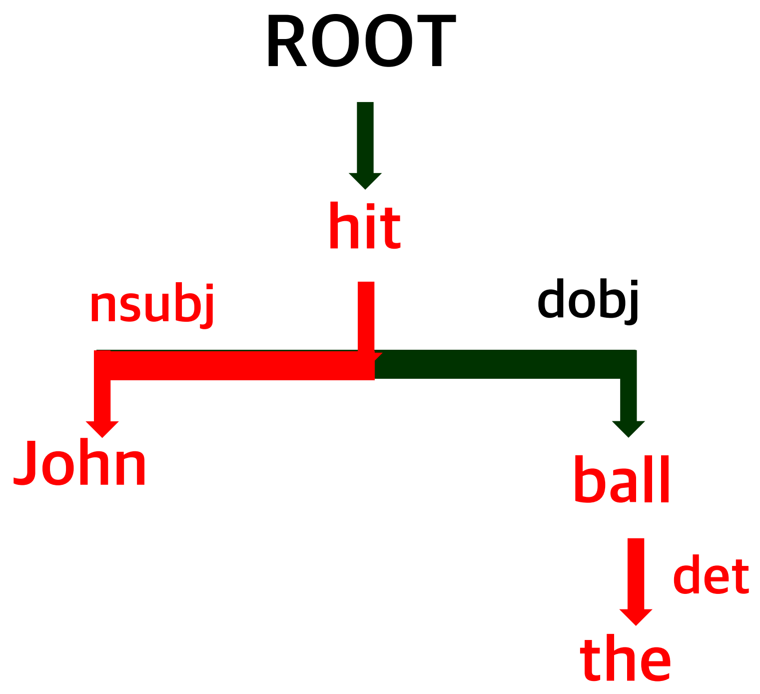
STACK (σ)	BUFFER (β)	Set of Arcs (A)
ROOT hit the ball	John hit the ball	(hit, nnsubj, John) (ball, det, the)

Decision Process

shift -> shift -> Left-Arc (nsubj) -> shift -> shift -> Left-Arc (det)

05 Dependency Parsing Method

#2 Transition-based Parsing



Input Sentence

“John hit the ball”

$c = (\sigma, \beta, A)$

State (c)

remove

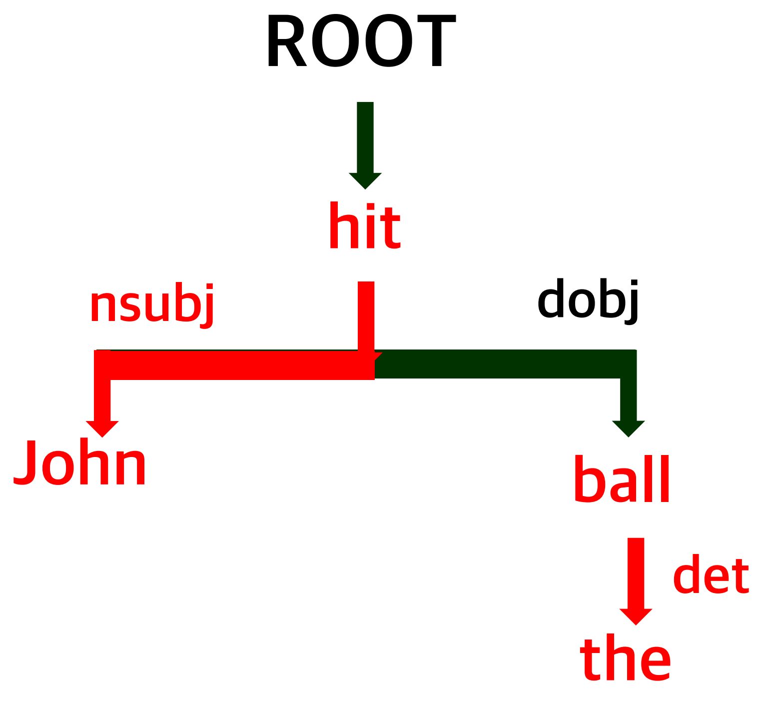
STACK (σ)	BUFFER (β)	Set of Arcs (A)
ROOT hit the ball	John hit the ball	(hit, nnsubj, John) (ball, det, the) (hit, dobj, ball)

Decision Process

shift -> shift -> Left-Arc (nsubj) -> shift -> shift -> Left-Arc (det) -> Right-Arc (dobj)

05 Dependency Parsing Method

#2 Transition-based Parsing



Input Sentence

"John hit the ball"

$c = (\sigma, \beta, A)$

State (c)

remove

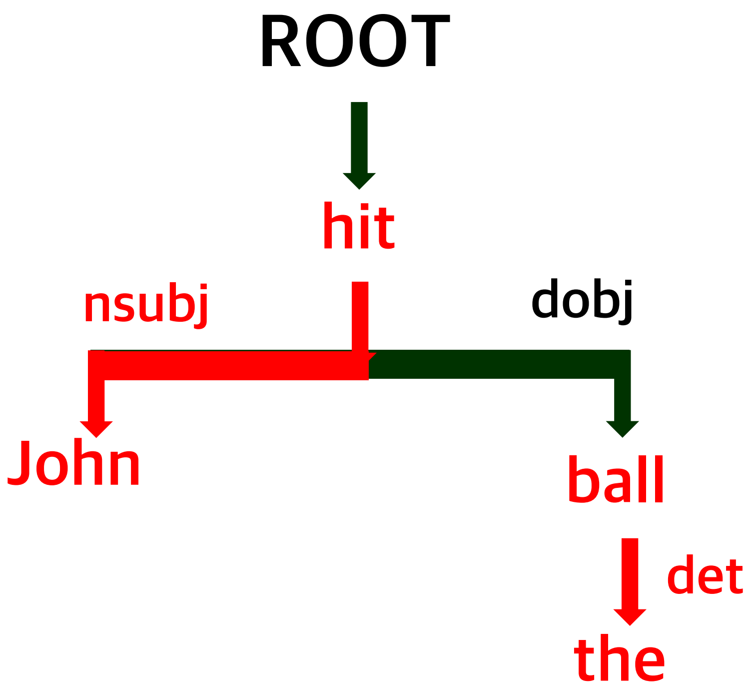
STACK (σ)	BUFFER (β)	Set of Arcs (A)
ROOT hit the ball	John hit the ball	(hit, nnsubj, John) (ball, det, the) (hit, dobj, ball) (ROOT, root, hit)

Decision Process

shift -> shift -> Left-Arc (nsubj) -> shift -> shift -> Left-Arc (det) -> Right-Arc (dobj) -> Right-Arc (dobj)

05 Dependency Parsing Method

#2 Transition-based Parsing



Input Sentence

"John hit the ball"

$c = (\sigma, \beta, A)$

f(c) ← State (c)

STACK (σ)	BUFFER (β)	Set of Arcs (A)
ROOT hit the ball	John hit the ball	(hit, nnsubj, John) (ball, det, the) (hit, dobj, ball) (ROOT, root, hit)

Decision Process

shift -> shift -> Left-Arc (nsubj) -> shift -> shift -> Left-Arc (det) -> Right-Arc (dobj) -> Right-Arc (dobj)

05 Dependency Parsing Method

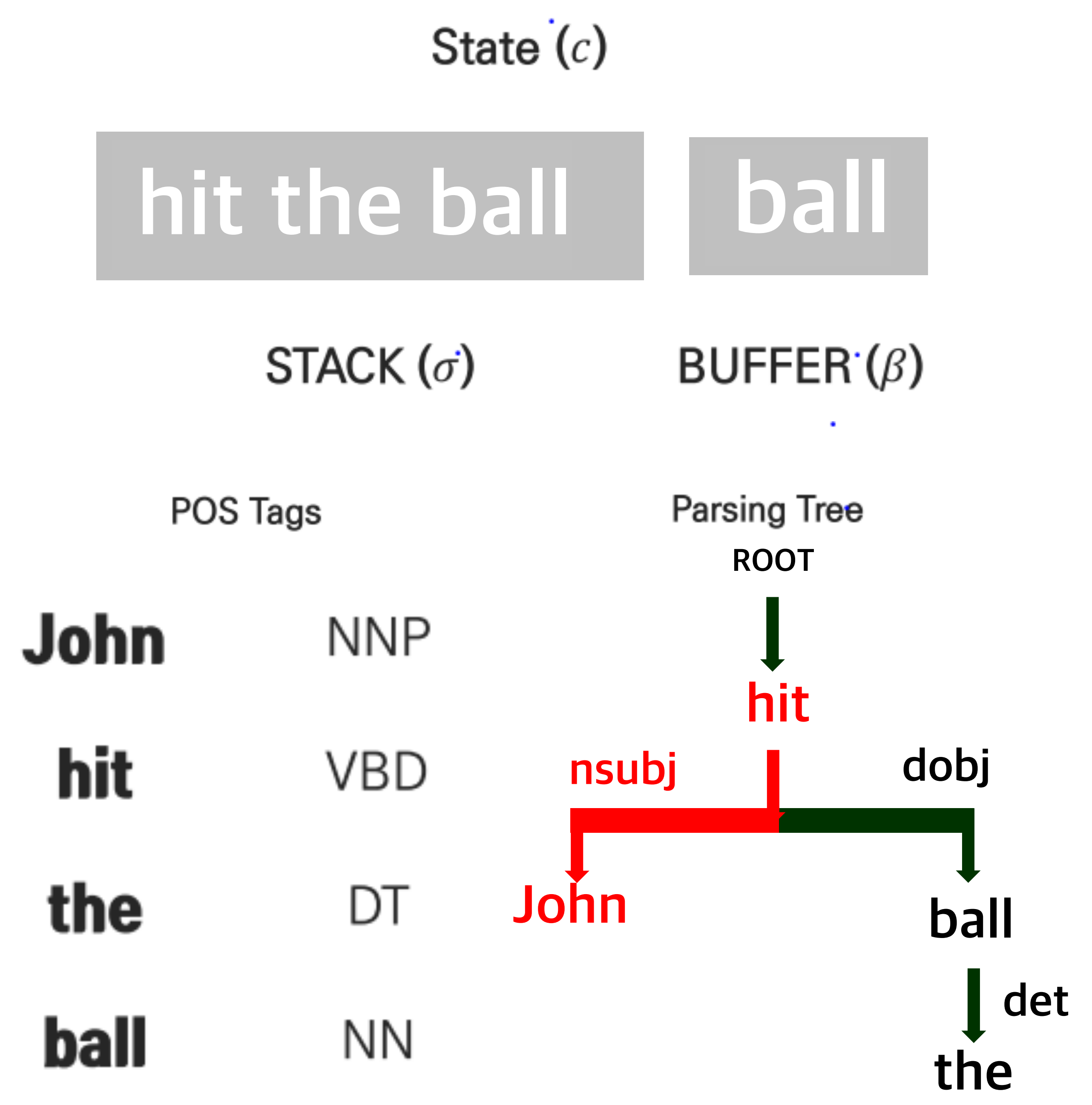
#3 Conventional Feature Representation

Notations

$s_1.w$	Stack의 첫번째 단어
$b_1.w$	Buffer의 첫번째 단어
$s_1.t$	Stack의 첫번째 단어의 POS Tag
$lc(s_1).w$	Stack의 첫번째 단어의 left-child 단어
$rc(s_1).w$	Stack의 첫번째 단어의 right-child 단어
$lc(s_1).t$	Stack의 첫번째 단어의 left-child 단어의 POS Tag
…	…

Example

$s_1.w$	$b_1.t$	$lc(s_2).w$	$rc(s_2).t$
the	NN	**John**	NULL



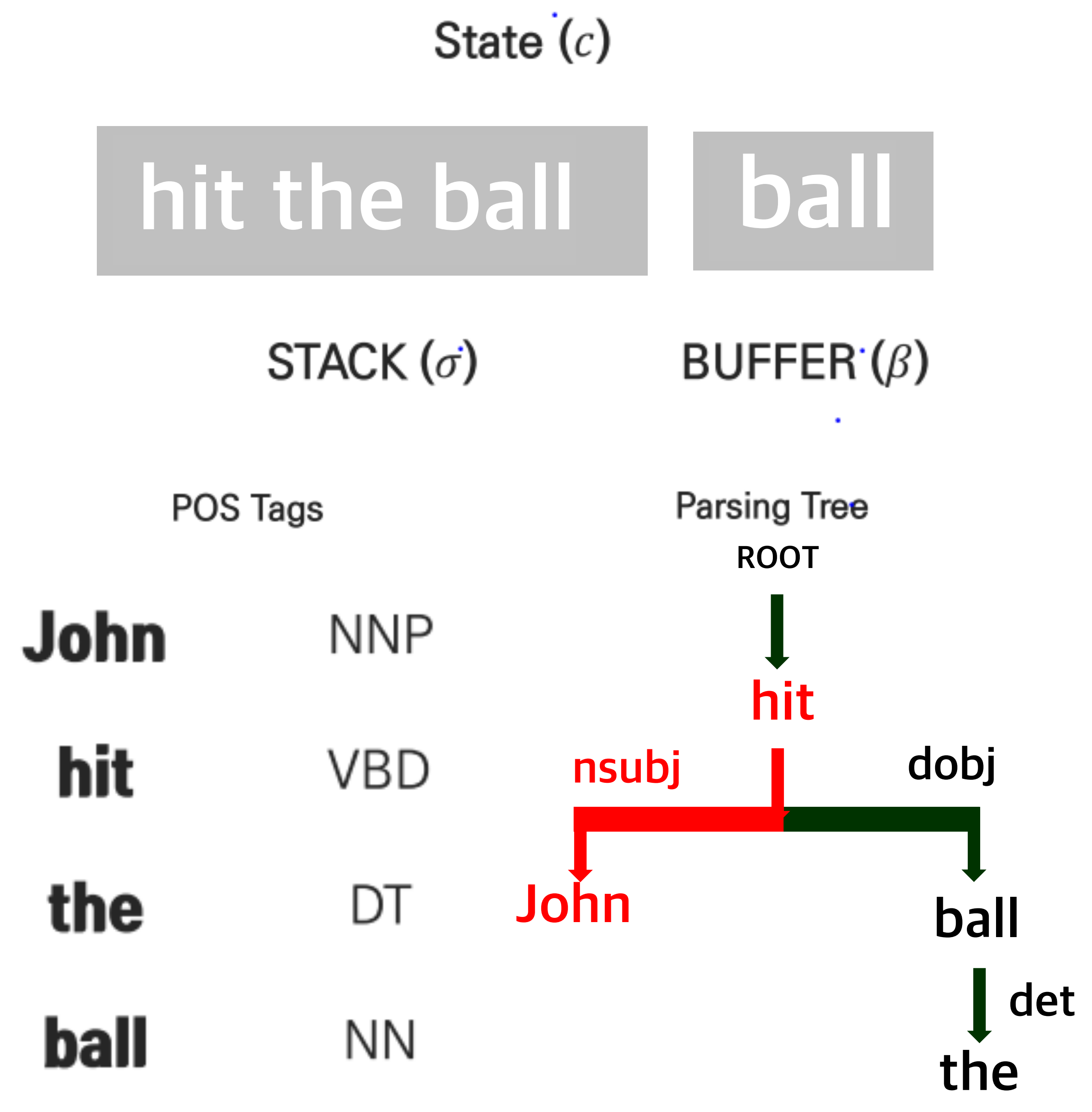
05 Dependency Parsing Method

#3 Conventional Feature Representation

Indicator Features

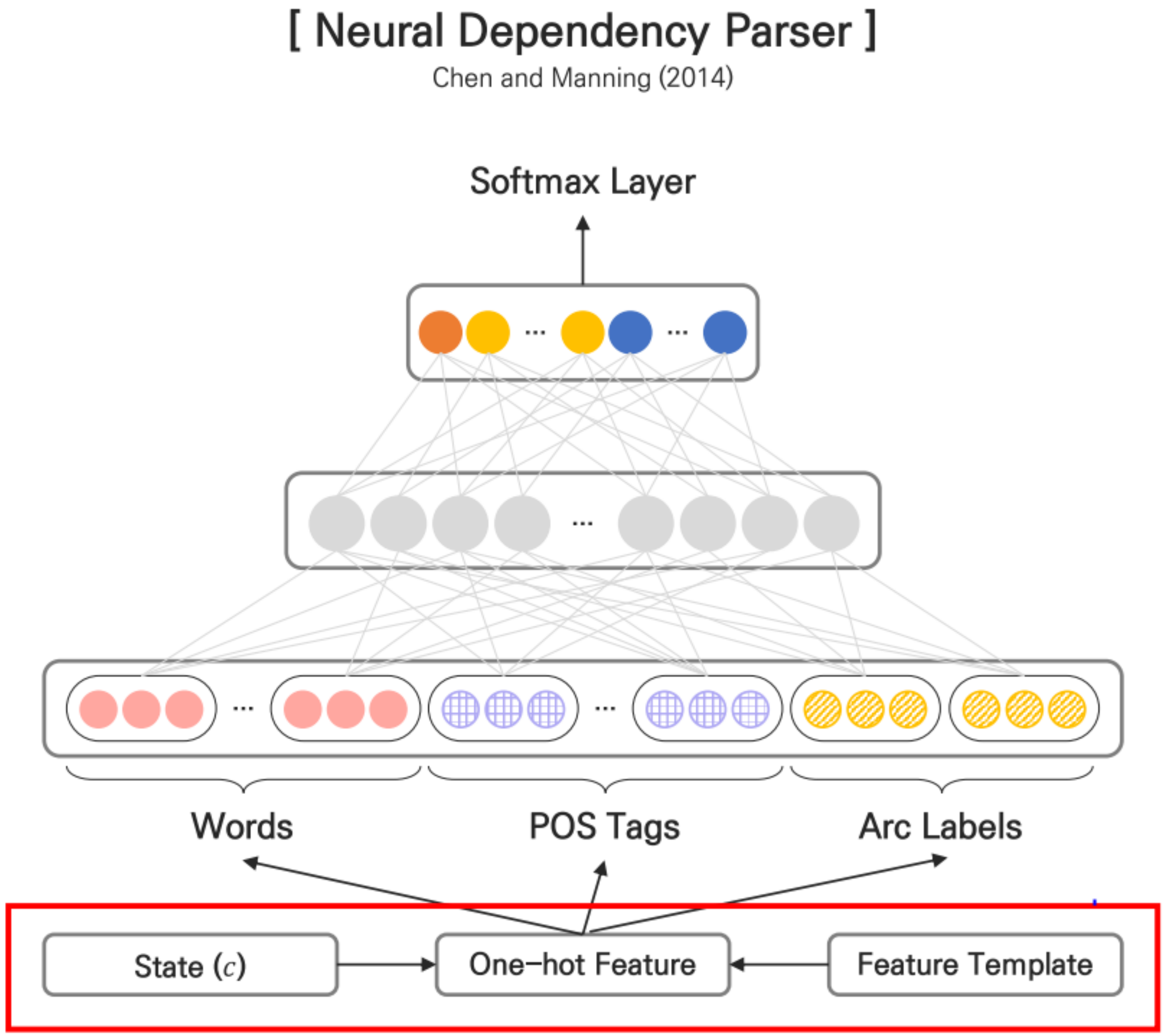
1	$s_1.w =$ **the** $\quad s_1.t =$ DT
0	$s_2.w =$ **hit** $\quad s_2.t =$ VBD $\quad b_1.t =$ DT
0	$lc(s_2).w =$ **John** $\quad lc(s_1).w =$ **hit**
1	$lc(s_2).w =$ **John** $\quad lc(s_1).t =$ NNP
…	…

- Binary & Sparse representation
- 일반적으로 1~3개 요소가 결합된 indicator features
- 계산비용 높음
- 단어 또는 POS Tag의 의미 반영 못함



05 Dependency Parsing Method

#4 Neural Dependency Parser



1. Feature Selection

05 Dependency Parsing Method

#4 Neural Dependency Parser

Feature Template

- STACK과 BUFFER의 top 3 단어 (6개) $s_1,\ s_2,\ s_3,\ b_1,\ b_2,\ b_3$

  [over, quick, ROOT, the, lazy, dog]

- STACK top 1, 2 단어의 1st and 2nd left and right child 단어 (8개)

  $lc_1(s_1), rc_1(s_1), lc_2(s_1), rc_2(s_1)$ $lc_1(s_2), rc_1(s_2), lc_2(s_2), rc_2(s_2)$

  [Null, Null, Null, Null] [fox, jumping, is, and]

- STACK top 1, 2 단어의 (left of left) and (right of right) child 단어 (4개)

  $lc_1(lc_1(s_1)), rc_1(rc_1(s_1)), lc_1(lc_1(s_2)), rc_1(rc_1(s_2))$

  [Null, Null] [the, Null]

- 선택된 word feature에 해당하는 POS Tag (18개)

  [IN(전치사), JJ(형용사), ROOT, DT(한정사), …, Null, DT(한정사), Null]

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

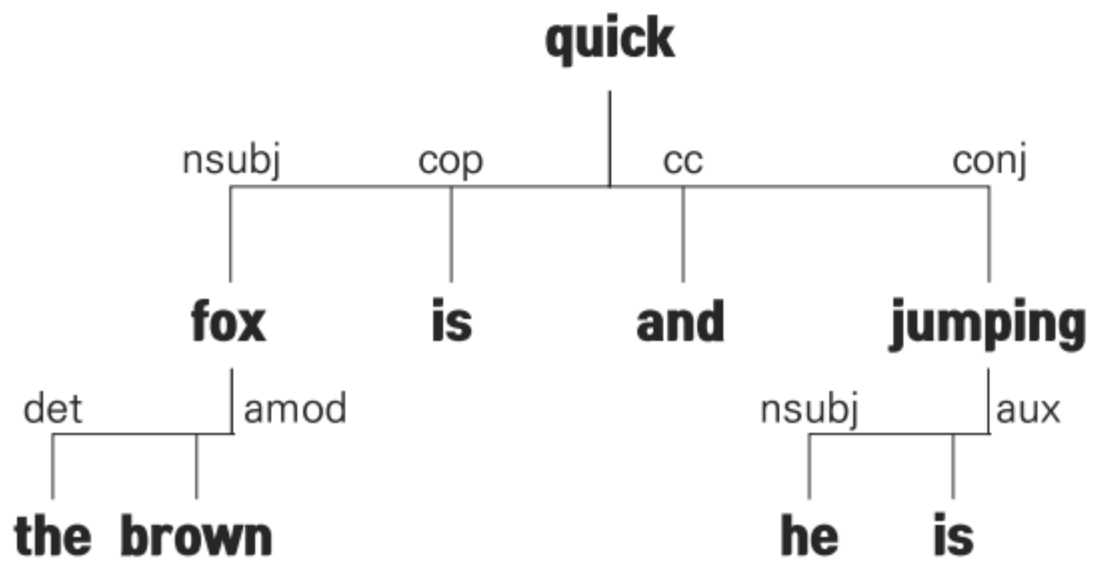
  [Null, Null, …, nsubj(주어), conj(접속사), cop(연결사), cc(등위), …, Null]

Example State (c)

Input Sentence

"The brown fox is quick and he is jumping over the lazy dog"

STACK (σ)	BUFFER (β)
ROOT quick over	the lazy dog



05 Dependency Parsing Method

#4 Neural Dependency Parser

One−hot Representation

Word Features

S^w = [over, quick, ROOT, the, ⋯, and, Null , Null, the , Null]

	Null	ROOT	and	the	⋯	over	quick
over	0	0	0	0	0	1	0
quick	0	0	0	0	0	0	1
ROOT	0	1	0	0	0	0	0
⋯				⋯			
Null	1	0	0	0	0	0	0
The	0	0	0	1	0	0	0
Null	1	0	0	0	0	0	0

$S'^w \in \mathbb{R}^{18\times N_w}$

POS Tag Features

S^t = [IN, JJ, ROOT, DT, JJ, ⋯, CC, Null, Null, DT, Null]

	Null	ROOT	DT	NN	⋯	JJ	VBD
IN	0	0	0	0	0	0	0
JJ	0	0	0	0	0	1	0
ROOT	0	1	0	0	0	0	0
⋯				⋯			
Null	1	0	0	0	0	0	0
DT	0	0	1	0	0	0	0
Null	1	0	0	0	0	0	0

$S'^t \in \mathbb{R}^{18\times N_t}$

Arc−label Features

S^l = [Null, Null, Null, ⋯, nsubj, conj, cop , cc, Null, ⋯]

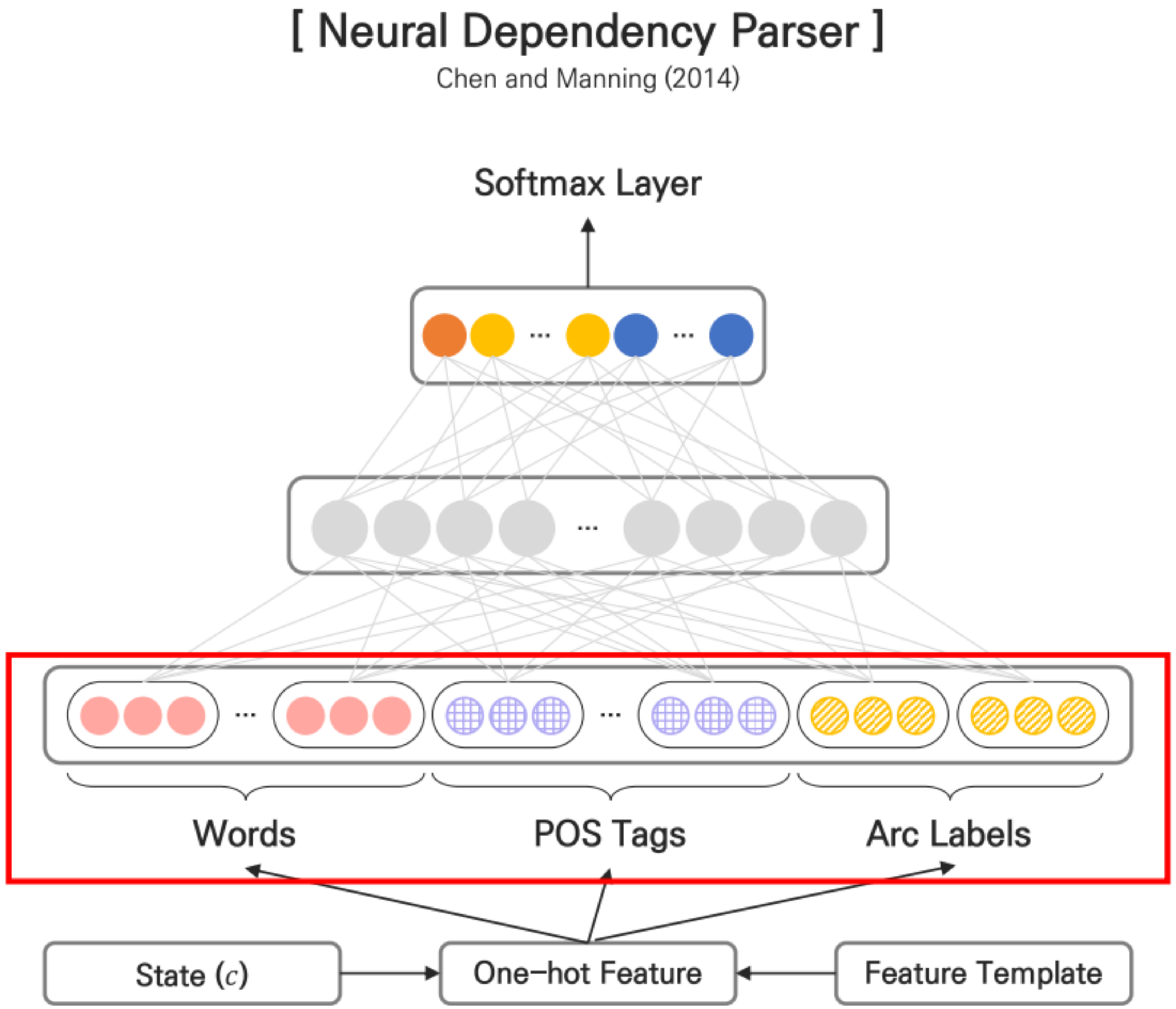
	Null	ROOT	nsubj	cc	⋯	cop	conj
Null	1	0	0	0	0	0	0
Null	1	0	0	0	0	0	0
Null	1	0	0	0	0	0	0
⋯				⋯			
nsubj	0	0	1	0	0	0	0
conj	0	0	0	0	0	0	1
cop	0	0	0	0	0	1	0

$S'^l \in \mathbb{R}^{12\times N_l}$

05 Dependency Parsing Method

#4 Neural Dependency Parser

2. Feature Embedding



05 Dependency Parsing Method

#4 Neural Dependency Parser

Feature Embedding

Word Features

	Null	ROOT	and	the	···	over	quick
over	0	0	0	0	0	1	0
quick	0	0	0	0	0	0	1
ROOT	0	1	0	0	0	0	0
···				···			
Null	1	0	0	0	0	0	0
The	0	0	0	1	0	0	0
Null	1	0	0	0	0	0	0

$$S'^{w} \in \mathbb{R}^{18 \times N_w}$$

×

Word Embedding Matrix

Null	0.2	0.1	0.7		0.7	1.2	0.1
ROOT	0.3	0.8	2.3		1.2	0.1	1.3
and	0.7	1.0	1.1		0.2	0.6	0.1
the	0.7	0.4	0.3		2.1	0.3	1.0
···				···			
over	0.3	0.2	0.5		1.0	0.2	0.7
quick	1.2	0.8	0.2		2.0	0.3	0.6

$$E^{w} \in \mathbb{R}^{d \times N_w}$$

=

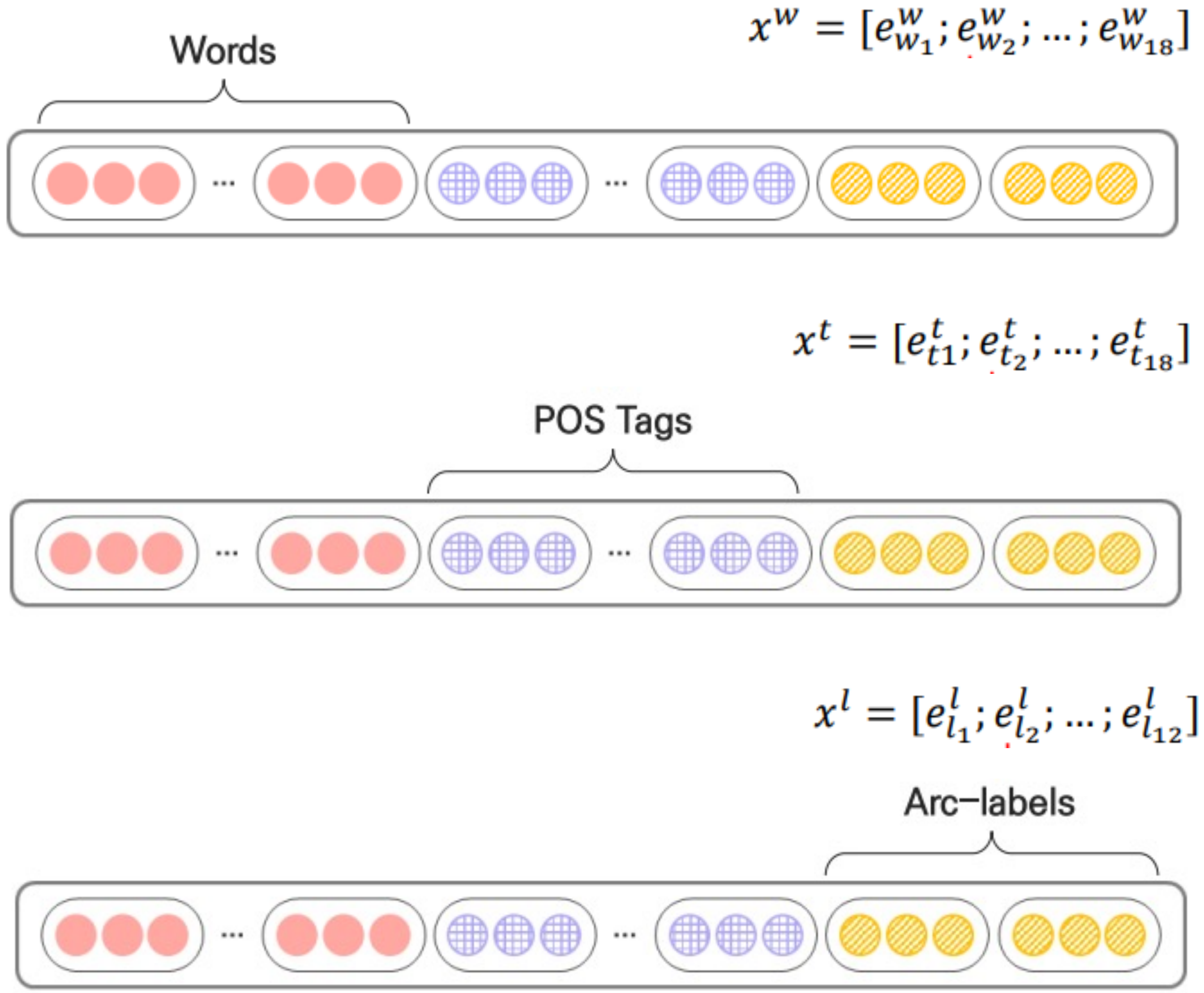
Embedded Matrix

$e^{w}_{w_1}$	0.3	0.2	0.5		1.0	0.2	0.7
$e^{w}_{w_2}$	1.2	0.8	0.2		2.0	0.3	0.6
$e^{w}_{w_3}$	0.3	0.8	2.3		1.2	0.1	1.3
				···			
$e^{w}_{w_{16}}$	0.2	0.1	0.7		0.7	1.2	0.1
$e^{w}_{w_{17}}$	0.7	0.4	0.3		2.1	0.3	1.0
$e^{w}_{w_{18}}$	0.2	0.1	0.7		0.7	1.2	0.1

$$x^{w} = [e^{w}_{w_1}; e^{w}_{w_2}; \dots; e^{w}_{w_{18}}]$$

05 Dependency Parsing Method

#4 Neural Dependency Parser



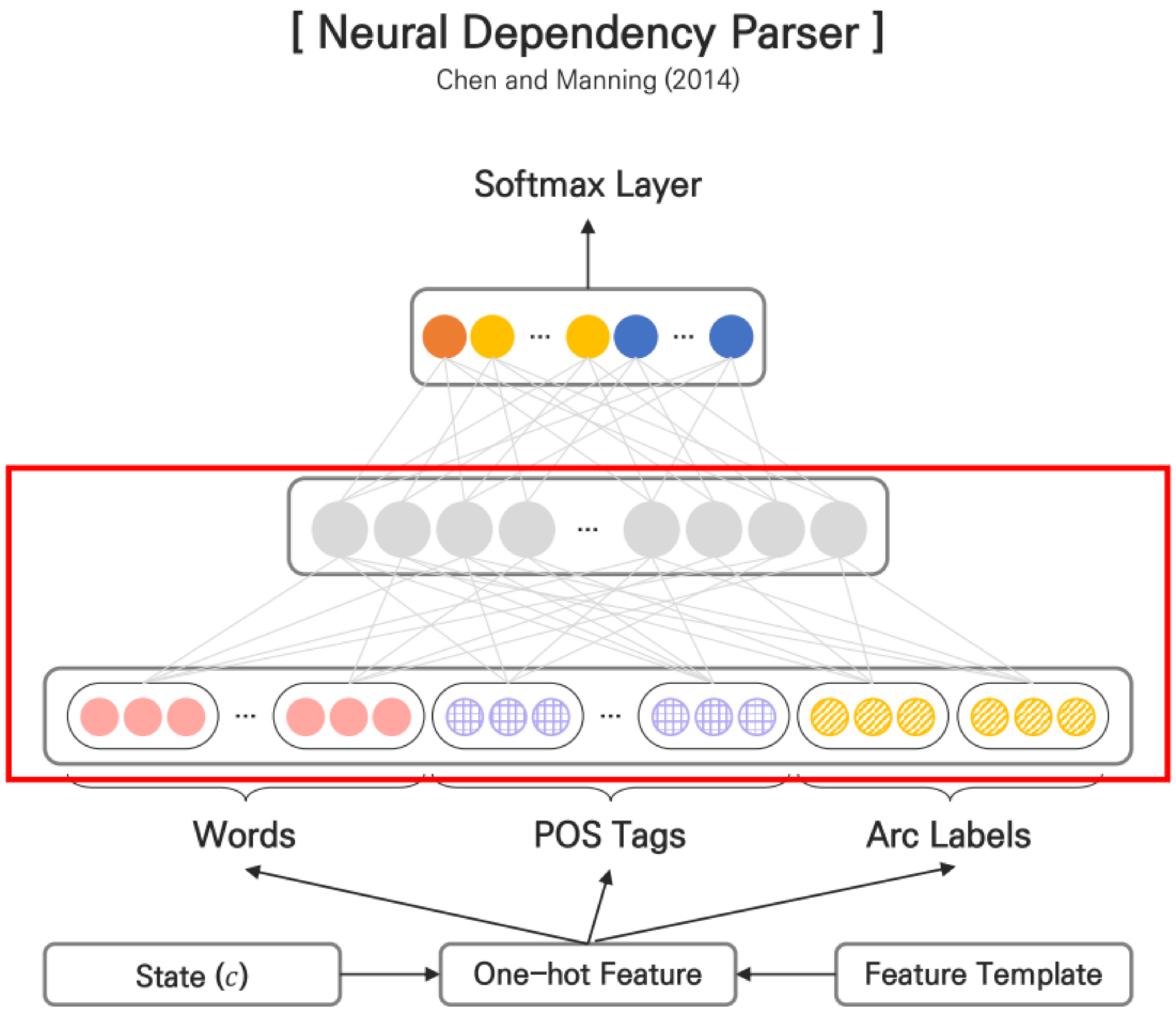
Embedded Matrix

$e^w_{w_1}$	0.3	0.2	0.5		1.0	0.2	0.7
$e^w_{w_2}$	1.2	0.8	0.2		2.0	0.3	0.6
$e^w_{w_3}$	0.3	0.8	2.3		1.2	0.1	1.3
				…			
$e^w_{w_{16}}$	0.2	0.1	0.7		0.7	1.2	0.1
$e^w_{w_{17}}$	0.7	0.4	0.3		2.1	0.3	1.0
$e^w_{w_{18}}$	0.2	0.1	0.7		0.7	1.2	0.1

05 Dependency Parsing Method

#4 Neural Dependency Parser

3. Hidden Layer

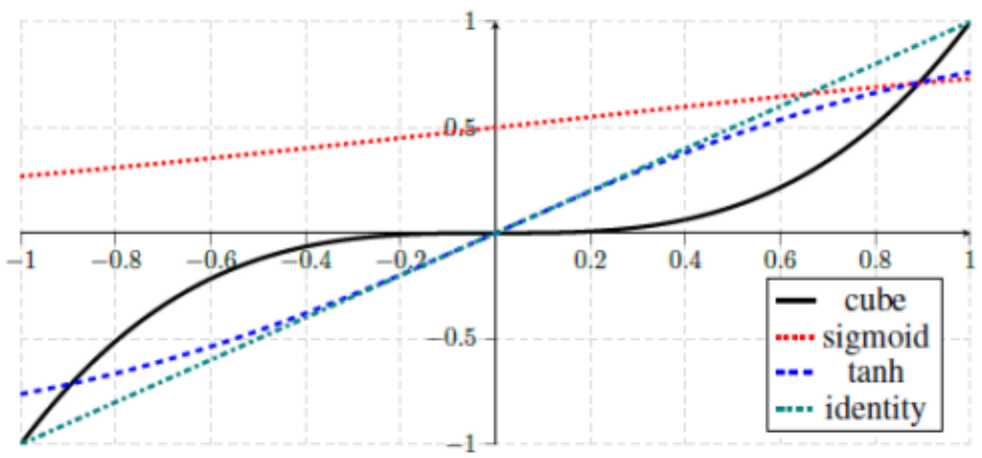


05 Dependency Parsing Method

#4 Neural Dependency Parser

Hidden Layer : 일반적인 feed forward network

- Embedding vector * weight matrix + bias vector
- ReLU, Sigmoid, Tanh와 같은 일반적인 activation function을 사용하지 않음
- word, POS tag, arc-label 간 상호작용을 반영할 수 있는 cube function을 사용함



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

$$= (w_1 x_1 + w_1 x_2 + \cdots + w_{48} x_{48} + b)^3$$

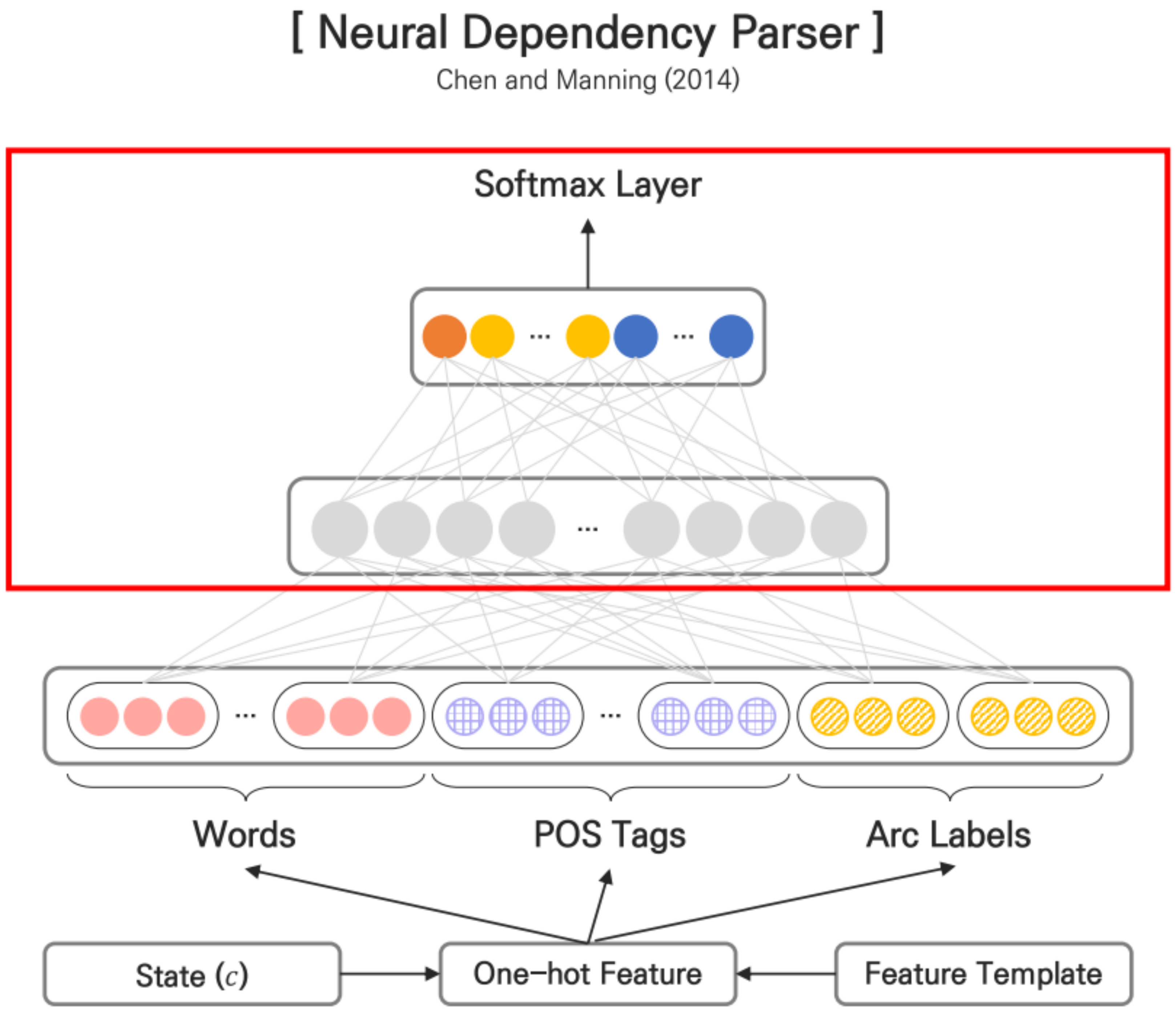
$$= \sum_{i,j,k} (w_i w_j w_k) x_i x_j x_k + \sum_{i,j} b(w_i w_j) x_i x_j + \cdots$$

각 word, POS tag, arc-label의 조합

05 Dependency Parsing Method

#4 Neural Dependency Parser

4. Softmax Layer



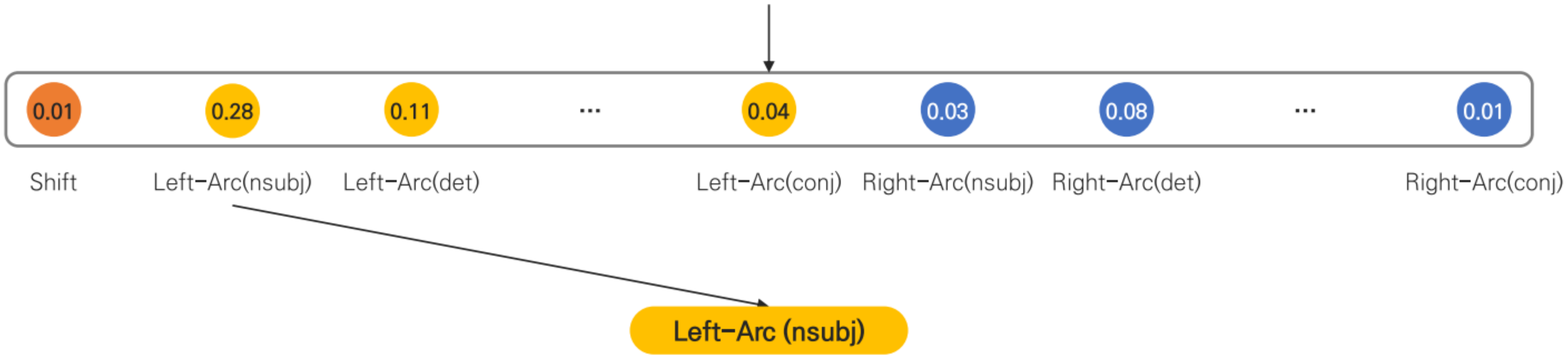
05 Dependency Parsing Method

#4 Neural Dependency Parser

Softmax Layer

- Hidden layer를 거친 feature vecto를 linear projection 후 softmax function 적용
- Shift, Left-Arc, Right-Arc 중 가장 확률값이 높은 경우의 수를 output으로 산출

$$p = softmax(W_2 h)$$



05 Dependency Parsing Method

#4 Neural Dependency Parser

Evaluation Measures

- Unlabeled Attachment Score (UAS): Arc 방향만 예측
  Shift Left-Arc Right-Arc

- Labeled Attachment Score (LAS): Arc 방향과 함께 label 예측
  Shift Left-Arc (nsubj) Left-Arc (det) … Right-Arc (det) …

Results (English Penn Treebank Datasets)

Parser	UAS	LAS	Sentence / sec.	비고
MaltParser (2007)	89.8	87.2	469	Transition–based Parser (Indicator Feature)
MSTParser (2007)	91.4	88.1	10	Graph–based Parser
TurboParser (2010)	**92.3**	89.6	8	Graph–based Parser
Chen and Manning (2014)	92.0	**89.7**	**654**	Transition–based Parser (Dense Feature)

05 Dependency Parsing Method

#4 Neural Dependency Parser

Ablation Studies

- Cube function이 타 activation function보다 높은 성능 기록
- Pre-trained word vector (Word2Vec)를 사용하는 것이 random initialization보다 더 높은 성능 기록
- Word, POS, label 정보 모두 활용하는 것이 가장 높은 성능 기록

POS and Label Embedding

- Random Initialization된 POS tag와 Arc-label vector가 학습이 진행되면서 의미적 유사성 내포
- t-SNE를 통해 2차원 공간상에 표현했을 때 유사한 요소들이 가까이 위치