

---

# 개체명 인식 (Named Entity Recognition)

---

건국대학교 컴퓨터공학부 /  
KAIST 전산학부 (겸직)

김학수

# 개체명 인식

Core of Information  
Extraction (Knowledge  
Extraction)

- 개체명 인식(Named Entity Recognition)
  - 문서 내에서 인명, 기관명, 지명, 시간, 날짜 등 고유한 의미를 가지는 단어 열을 추출하여 범주를 결정하는 것

NC는 25일 18시 30분에  
서울 잠실구장에서 시작한  
2016 타이어뱅크 KBO리그  
플레이오프(5전3선승제) 4  
차전에서 박석민의 역전  
홈런포에 힘입어 LG를 8-3  
으로 제압했다.

• 사람 • 지역 • 기관 • 날짜 • 시간



# Named Entity Recognition (NER)

---

- Finding named entities in a text → Segmentation
  - Classifying them to the corresponding classes → Classification
  - Assigning a unique identifier from a database → Grounding
- 
- „Steven Paul Jobs, co-founder of Apple Inc, was born in California.“
  - „Steven Paul Jobs, co-founder of Apple Inc, was born in California.“
  - „Steven Paul Jobs [PER], co-founder of Apple Inc [ORG], was born in California [LOC].“
  - „Steven Paul Jobs [Steve\_Jobs], co-founder of Apple Inc [Apple\_Inc.], was born in California [California].“

# Named Entity Classes

---

- Person
  - Person names
- Organization
  - Companies, Government, Organizations, Committees, ..
- Location
  - Cities, Countries, Rivers, ..
- Date and time expression
- Measure
  - Percent, Money, Weight, ...
- Book, journal title
- Movie title
- Gene, disease, drug name



# NER Ambiguity

---

- Ambiguity between named entities and common words
  - May: month, verb, surname
  - Genes: VIP, hedgehog, deafness, wasp, was, if
- Ambiguity between named entity types
  - Washington (Location or Person)



# NER Task

---

- Similar to a classification task → Sequence Labeling Problem
  - Feature selection
  - Algorithms



# Gazetteer Features for NER

---

- 개체명 사전 자질
  - 데이터에서 개체명을 가지는 단어들을 사전 형태로 구축
    - 인명에 대한 사전 = {홍길동, 승주, 장영실, ...}
    - 지명에 대한 사전 = {서울, 대한민국, 춘천, ...}
    - 기관명에 대한 사전 = {한국기생충박람회, 한국특허정보원, ...}
  - 입력 문장 중 어절 또는  $n$ -gram에 대한 개체명 사전 포함 여부(0, 1)를 자질로 사용
  - 직접적인 개체명 정보를 주기 때문에 개체명 인식 성능 향상에 중요한 역할



# Quick Review: RNN 응용 구조

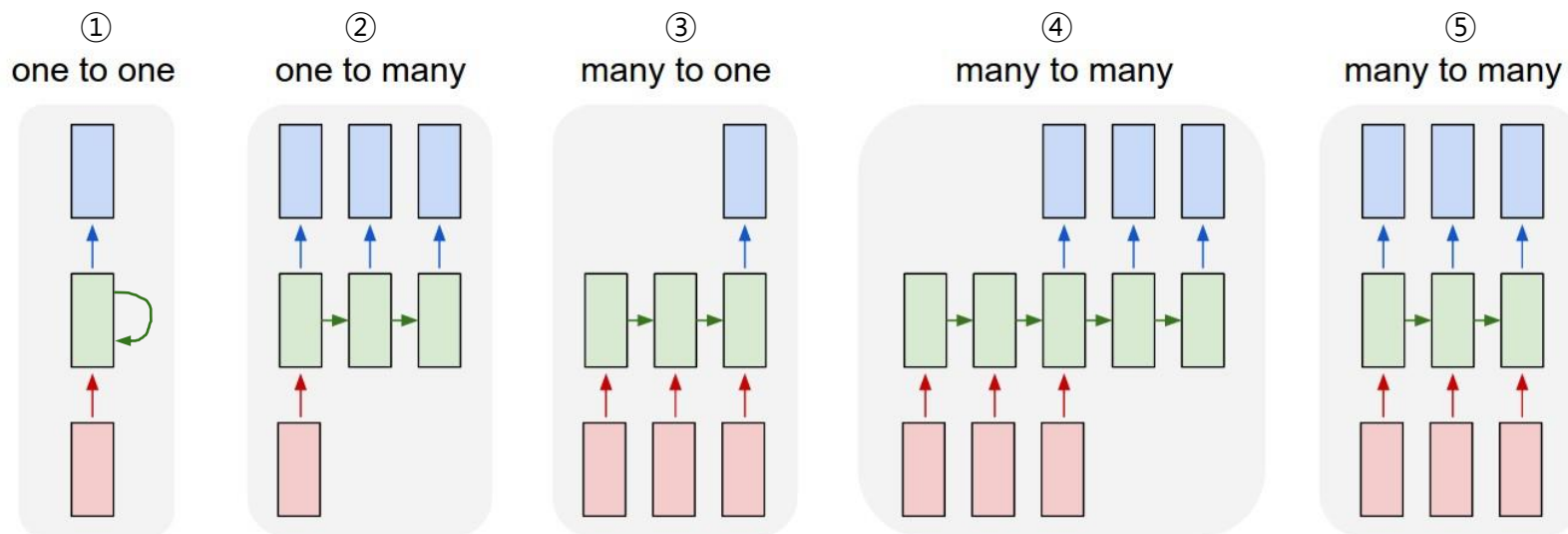
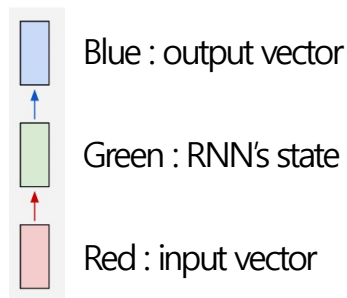


그림 참조: Stanford Lecture CS2031N, 2017  
(Fei-Fei Li & Justin Johnson & Serena Yeung)



## [활용 예]

- ① 일반적인 NN (FNN과 유사함)
- ② 자막 생성 (image captioning)
- ③ 분류 및 예측 (prediction)
- ④ 기계 번역
- ⑤ 순차 표지 부착

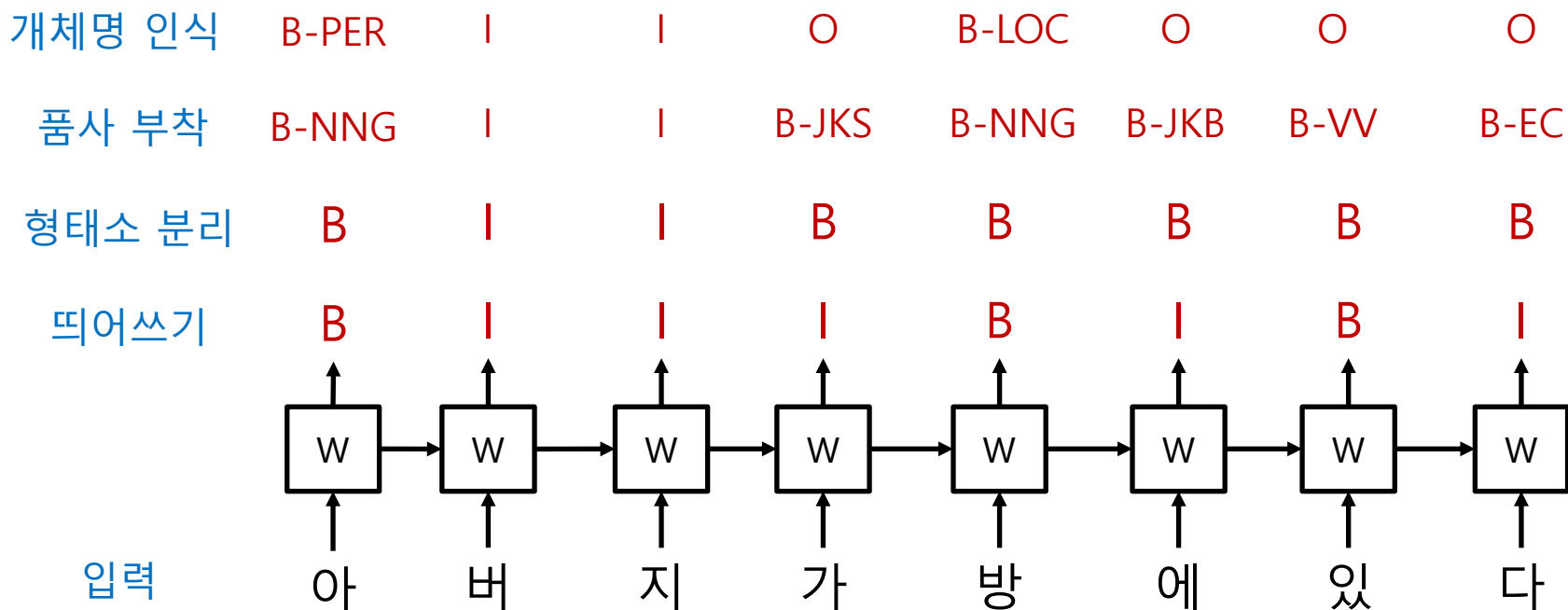




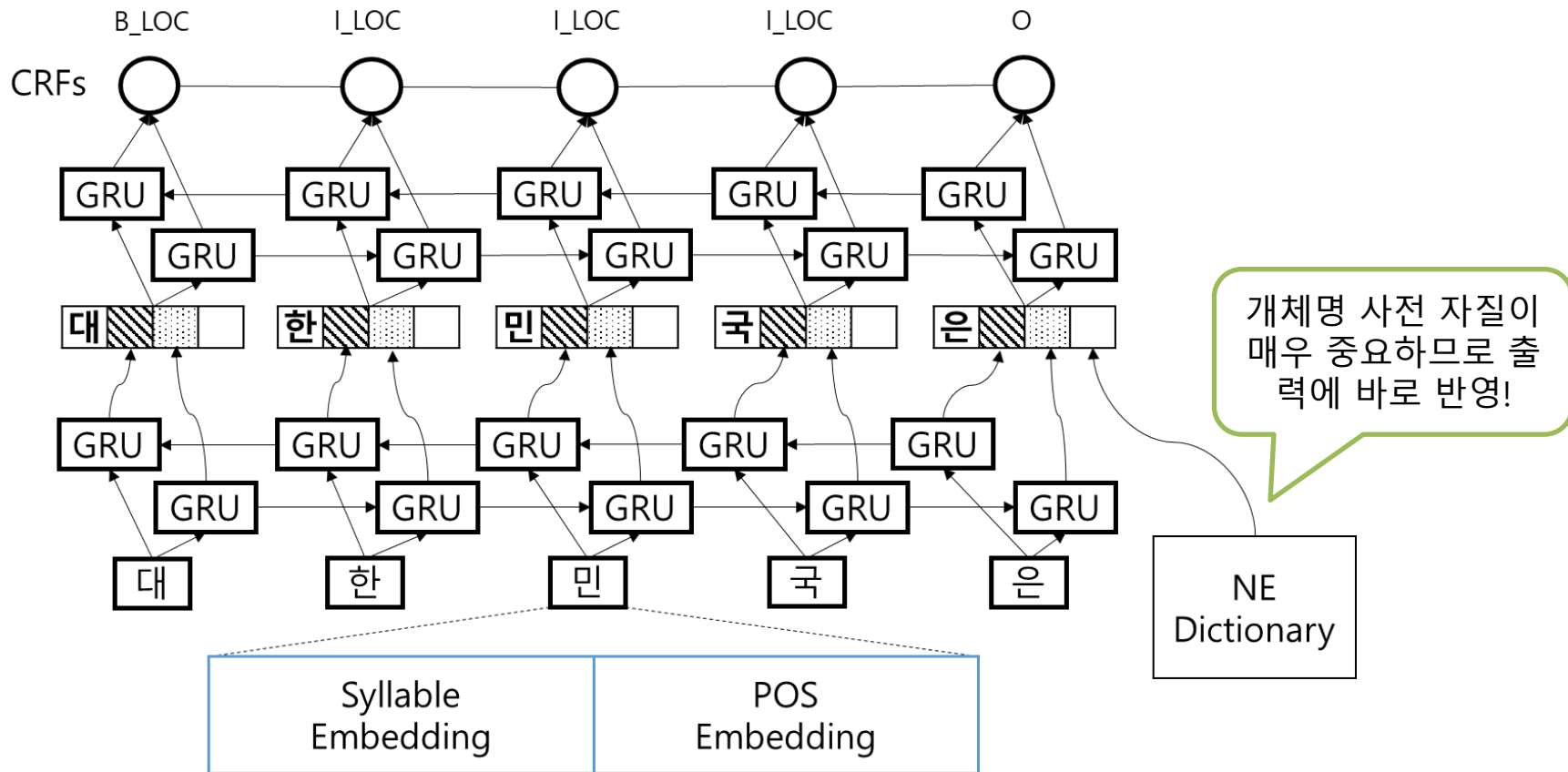
# Many-to-Many Model (Sequence Labeling)

- 순차적 레이블 부착
  - 연속된 입력에 대해 문맥을 반영하여 분류를 수행하는 것

BIO Notation for Segmentation: B(Beginner), I(Inner), O(outside)



# BiLSTM-CRFs for NER



# 실험 및 평가

- 실험 말뭉치

- 2016 국어정보처리시스템 개체명 말뭉치(약 5,000 문장)
- 인명, 지명, 기관명, 날짜, 시간의 5개의 개체 클래스

Description	Numbers
Person	3,416
Location	2,611
Organization	4,010
Date	2,688
Time	388



# 실험 및 평가

- 개체명 사전 자질 실험 결과
  - BiGRU-CRFs는 단층, Stacked BiGRU-CRFs는 2개의 계층구조
  - 개체명 사전 자질은 기존과 동일한 방법으로 하위 계층(입력단)에 사용한 성능

Model	Precision	Recall	F1-score
BiGRU-CRFs	0.8184	0.7022	0.7554
BiGRU-CRFs + 개체명 사전 자질	0.8655	0.7695	0.8147
Stacked BiGRU-CRFs	0.8328	0.7305	0.7783
Stacked BiGRU-CRFs + 개체명 사전 자질	0.8800	0.7871	0.8309



# 실험 결과

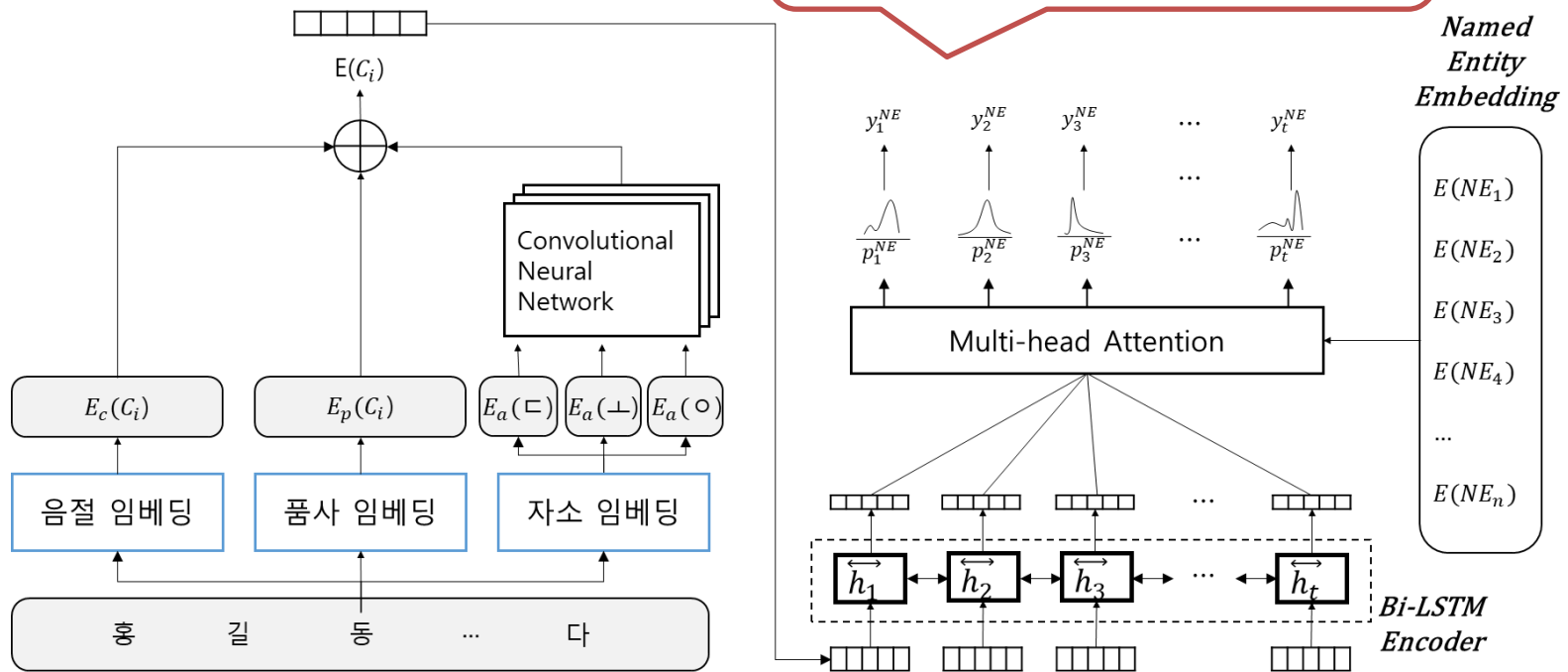
- 상위 계층에 개체명 사전 자질을 사용한 경우 성능 비교
  - Precision은 거의 동일하지만 Recall은 큰 폭으로 증가(5.12%)
  - Stacked BiGRU-CRFs에서 상위 계층에 개체명 사전 자질을 반영하는 것이 보다 효과적

사전 자질 사용 방법	Precision	Recall	F1-score
하위 계층 반영 방법	0.8800	0.7871	0.8309
상위 계층 반영 방법	0.8778 (- 0.22%)	0.8383 (+ 5.21%)	0.8576 (+ 2.67%)



# BiLSTM+LAN 모델

CRFs는 레이블 종류가 많아지면 속도 저하  
→ LAN을 이용하여 속도 및 성능 개선!



# 실험 결과

Model	Precision	Recall	F1-score
BiGRU-CRFs	0.8184	0.7022	0.7554
BiGRU-CRFs + 개체명 사전 자질	0.8655	0.7695	0.8147
Stacked BiGRU-CRFs	0.8328	0.7305	0.7783
Stacked BiGRU-CRFs + 하위 개체명 사전 자질	0.8800	0.7871	0.8309
Stacked BiGRU-CRFs + 상위 개체명 사전 자질	0.8778	0.8383	0.8576
BiLSTM+LAN			0.8616

속도 1.6배 향상  
(개체명 타입 수가 증가하면 속도 개선폭 증가)

# Fine-grained NER 필요성

---

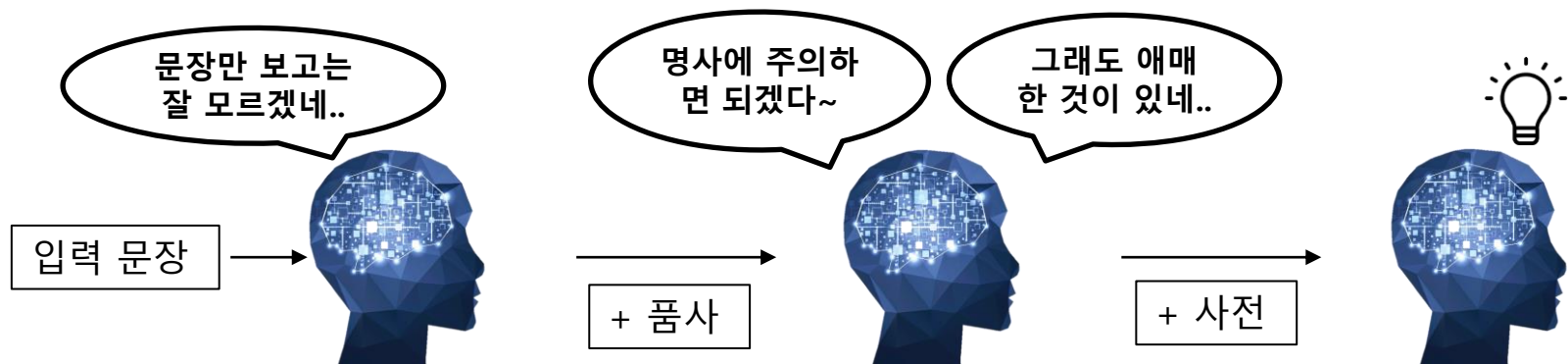
- 연구 배경

- 기존 인명, 지명, 기관명, 날짜, 시간의 클래스로는 정보 추출에 한계가 존재
  - 일반적으로 개체명 클래스가 많아질수록 성능 하락 현상이 일어남
- 상용화 가능한 성능을 보이는 세부 분류 개체명 인식이 필요
  - 계층적 인코딩 구조를 통해 언어 자질을 효율적으로 반영
  - 계층적 레이블링 구조를 통해 하위 대분류 개체명 정보를 상위 소분류 개체명 인식에서 활용



# 인코딩 계층 설계

- 인간의 개체명 인식 방법을 모방하여 **계층적 인코딩 구조** 설계
  - 문장을 읽는다.
  - 명사인 것 중에 개체명을 뽑는다.
  - 애매한 것은 사전을 참고한다.



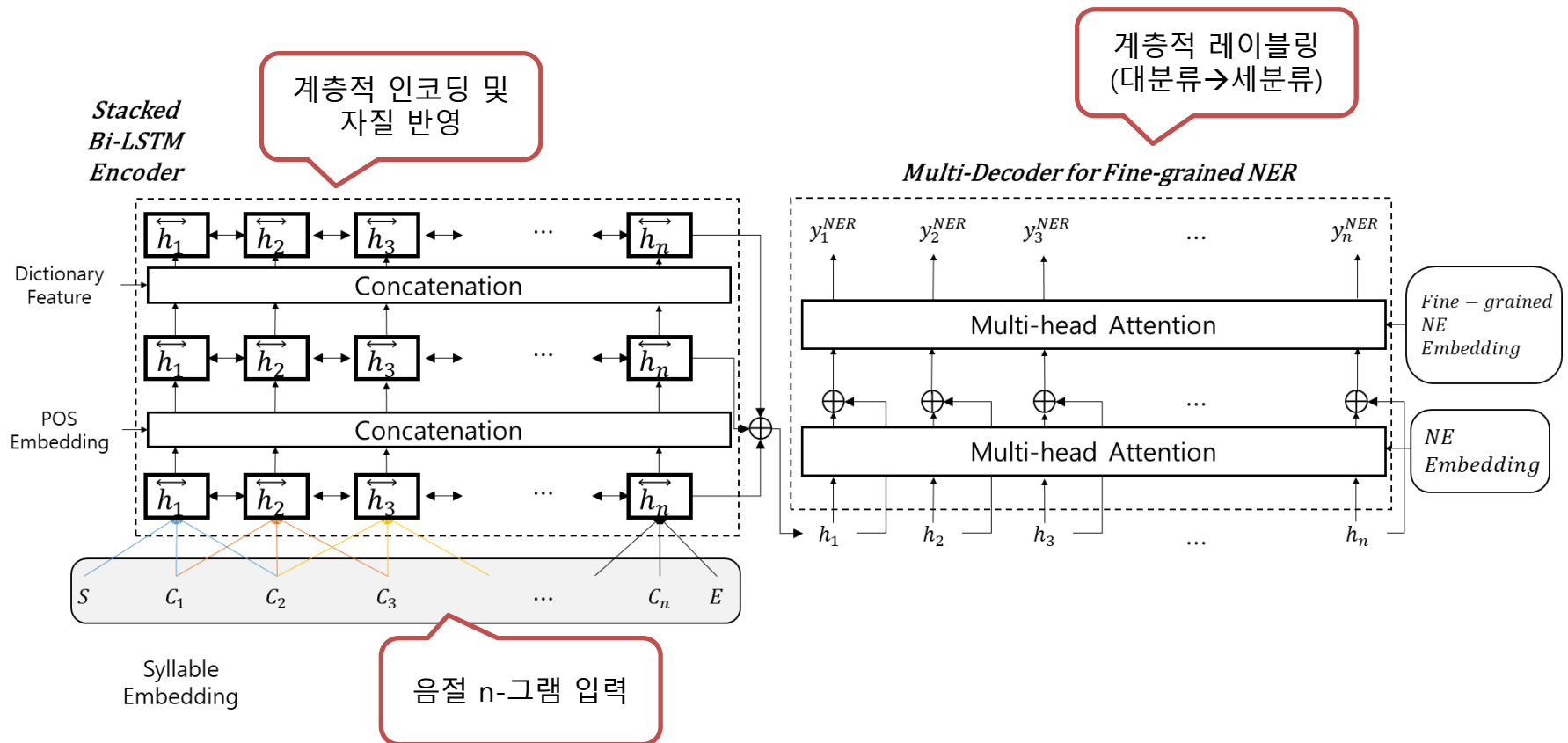
# 레이블링 계층 설계

- 대분류 범주에 속하는 세부 분류 개체명이 존재

Word	<b>Paris, London, Seoul, ...</b>
Coarse-grained	<b>Location</b>
Fine-grained	<b>Location_Capitalcity</b>

- ➔ 세부 분류 수행하기 전에 대분류 정보를 주면 도움 되지 않을까?
- ➔ 계층적 레이블링 구조를 적용하여 대분류 정보를 세부 분류에 반영

# Fine-grained NER 구조도



# 실험 및 평가

- 실험 데이터

- 일반 도메인 데이터 106,228 문장
  - 학습 데이터: 95,688개, 평가 데이터 10,600개
- 범주 구성
  - 대분류 15개
  - 세부 분류 147개
  - 예시

대분류	세부 분류
동물	조류, 어류, 포유류 등..
날짜	연도, 월, 일, 계절 등..

# 실험 및 평가

- 147개 세분류 실험 결과

Model	Precision	Recall	F1-score
Bi-LSTM + Single-decoder	0.783	0.681	0.728
Bi-LSTM + Multi-decoder	0.808	0.701	0.750
Bi-LSTM + Multi-decoder + Tri-gram Syllable	0.831	0.730	0.777
Stacked Bi-LSTM + Single-decoder + Tri-gram Syllable	0.844	0.752	0.795
Proposed Model	<b>0.865</b>	<b>0.769</b>	<b>0.814</b>

# 중첩 개체명 인식

- 중첩 개체명
  - 하나의 개체명 내에 다른 개체명이 포함된 경우
  - 최근 중첩 개체명 인식을 해결하려는 연구가 활발히 진행

that ' s just one of the ten sniper shootings *here in the greater Washington area* .

LOC

GPE

GPE

*Over 4000 guests from home and abroad* attended the opening ceremony .

LOC

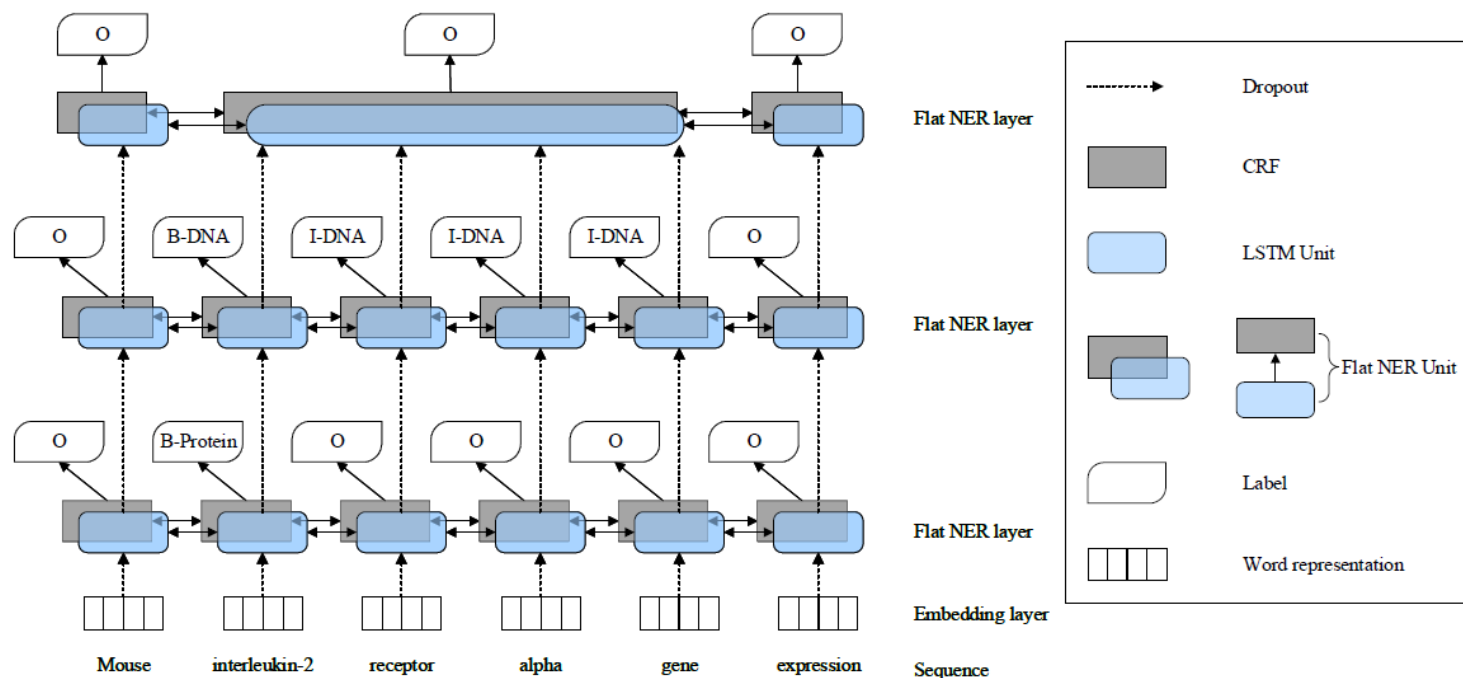
PER



# 순차 표지부착 기반 중첩 개체명 인식

- 다중 스택 개체명 인식

- 순차 레이블링 계층을 여러 층 쌓아서 다중 개체명 인식을 수행
- 가장 안쪽부터 바깥쪽까지 단계적으로 개체명 인식을 수행



# 순차 표지부착 기반 중첩 개체명 인식

- 다중 스택 모델의 문제점
  - 중첩 단계 만큼 계층을 쌓아야 하므로 중첩 단계가 높아질수록 많은 메모리를 소모
  - 학습 데이터에서 중첩 개체명의 비율이 적음 → 희소 데이터 문제
    - 중첩 개체명의 비율은 일반적으로 20~30%
    - 중첩 단계별 학습 레이블 불균형 문제로 이어짐
    - 상위 계층 일수록 성능이 급격히 하락하는 문제가 발생

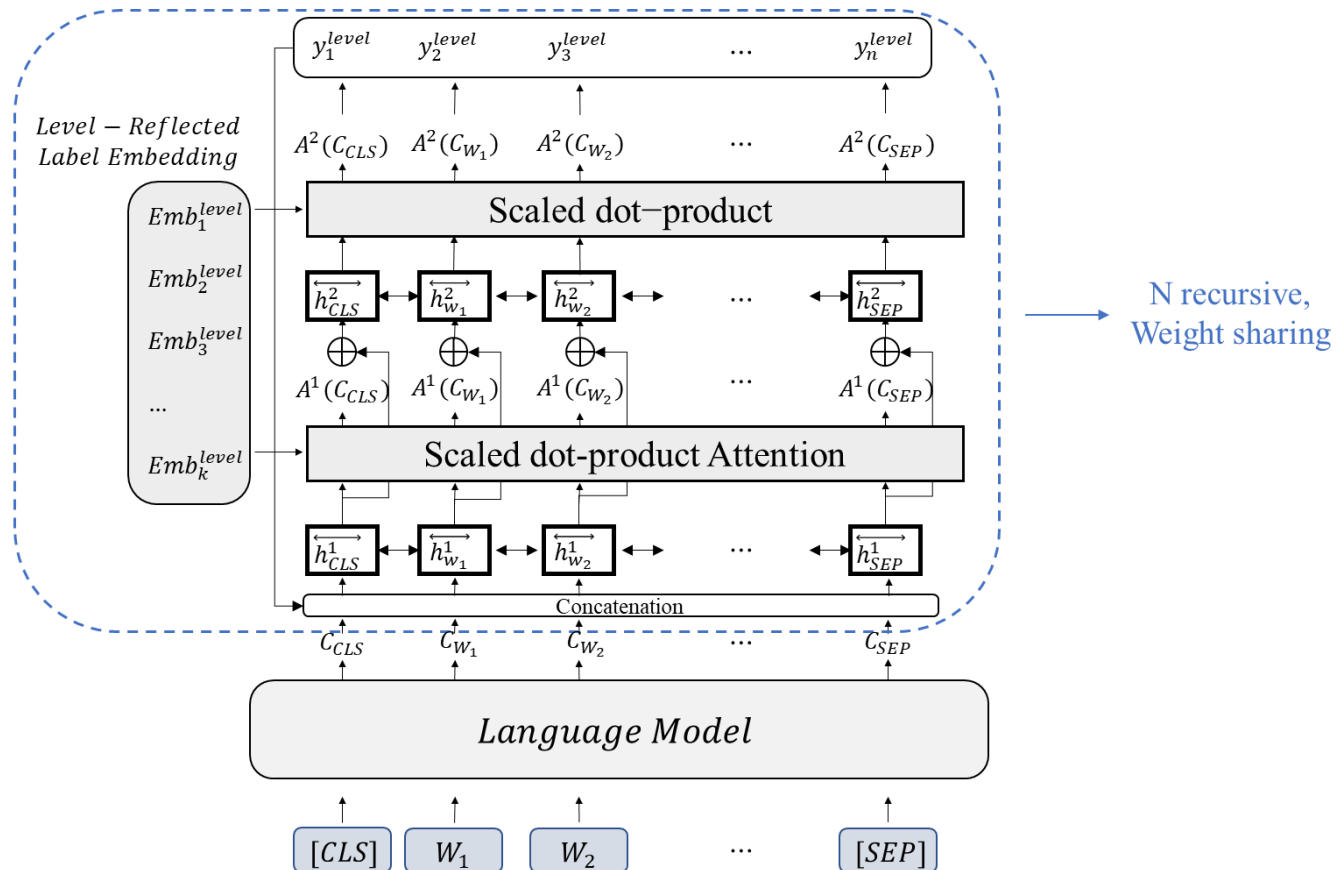
Layer	P (%)	R (%)	F (%)
Layer 1	72.86	69.82	71.31
Layer 2	56.88	27.59	37.15
Layer 3	0.00	0.00	0.00





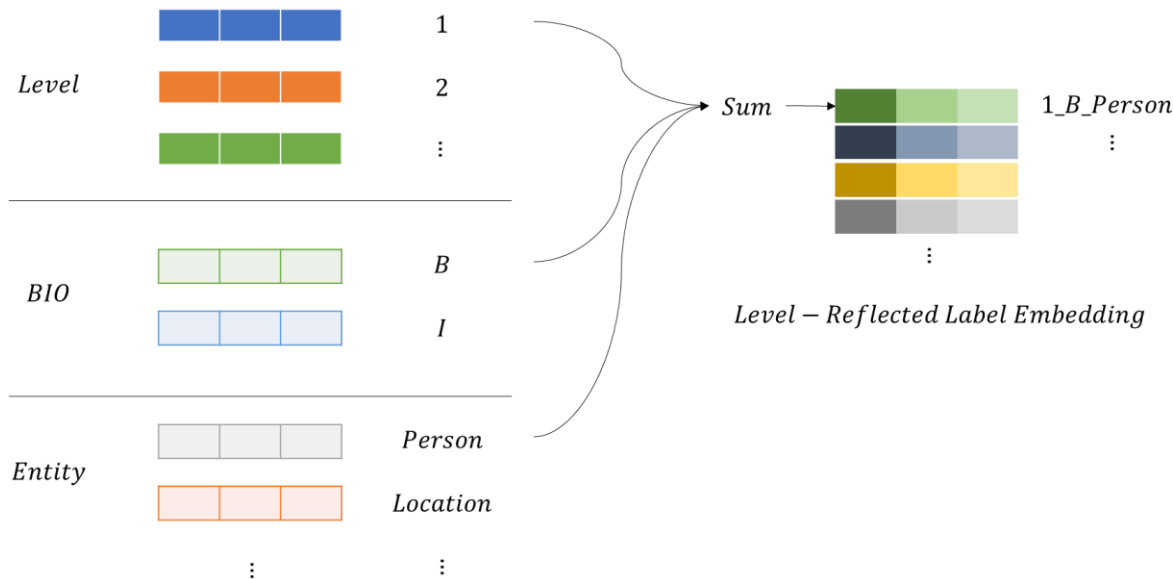
# 중첩 개체명 인식 모델

- RLAN(Recursive Label Attention Network)



# Level-Reflected Label Embedding

- Level-Reflected Label Embedding (LRLE)
  - 중첩 단계별 레이블 분포 불균형 문제를 완화
  - 레이블 구성요소들의 합으로 레이블 임베딩을 구성
  - 1단계에서 사용했던 개체 임베딩을 2단계 이상에서도 공유



# Inner Entity Pre-training Strategy

---

- Inner Entity Pre-training Strategy (IEPS)
  - 안쪽 개체명 인식의 오류가 다음 단계에 전파될 수 있음
  - 오류 전파를 최소화하기 위해 2단계 학습 방법을 적용
  - 1단계
    - 가장 안쪽 개체명만 학습
    - 이 단계에서는 모델이 재귀적으로 학습하지 않음
  - 2단계
    - 가장 안쪽 개체명부터 바깥쪽까지 모두 재귀적으로 학습

# 실험 및 평가 (1/4)

---

- Datasets
  - ACE 2004, ACE 2005
    - 7개의 개체 타입
    - 중첩 개체명 비율: 24%, 22%
  - GENIA
    - 5개의 개체 타입(DNA, RNA, Protein, Cell type, Cell line)
    - 중첩 개체명 비율: 10%



# 실험 및 평가 (2/4)

- Implementation Details
  - 3가지 언어 모델 각각 실험
    - BERT large
    - ELECTRA large
    - RoBERTa large
  - Parameter Settings

Hyper parameters	Value
Label embedding size	512
LSTM hidden size	256
Drop out	0.1
Learning rate	0.001



# 실험 및 평가 (3/4)

- 성능 비교

Model	F1-scores		
	ACE 2004	ACE 2005	GENIA
BERT-Seq2Seq	84.40	84.33	78.31
BERT-biaffine	86.70	85.40	80.50
BERT-MRC	85.98	86.88	<b>83.75</b>
BERT-RLAN (Our)	86.78	86.92	83.50
ELECTRA-RLAN (Our)	<b>87.11</b>	<b>87.19</b>	83.73
RoBERTa-RLAN (Our)	86.82	87.06	83.44

# 실험 및 평가 (4/4)

- Ablation Test
  - IEPS : 2단계 학습 방법
  - LRLE : 중첩 단계 반영 레이블 임베딩

GENIA			ACE 2004		
	F1-scores	$\Delta$		F1-scores	$\Delta$
ELECTRA-RLAN	83.73		ELECTRA-RLAN	87.11	
w/o IEPS	82.61	-1.12	w/o IEPS	85.94	-1.17
w/o LRLE	82.33	-1.40	w/o LRLE	85.86	-1.25
w/o IEPS, LRLE	81.19	-2.54	w/o IEPS, LRLE	84.71	-2.40

# Ranks in ACE & GENIA (with Paper)

<https://paperswithcode.com>

## ACE 2004

Rank	Model	F1 ↑	Paper	Code	Result	Year
1	Ours: cross-sentence ALB	90.3	<a href="#">A Frustratingly Easy Approach for Entity and Relation Extraction</a>	<a href="#">GitHub</a>	<a href="#">Paper with Code</a>	2020
2	Biaffine-NER	86.7	<a href="#">Named Entity Recognition as Dependency Parsing</a>	<a href="#">GitHub</a>	<a href="#">Paper with Code</a>	2020

MRC-based Model  
using ALBERT-xxlarge  
(NAACL 2021)

KU\_NLP  
(87.11)

## ACE 2005

Rank	Model	F1 ↑	Extra Training Data	Paper	Code	Result	Year
1	Ours: cross-sentence ALB	90.9	×	<a href="#">A Frustratingly Easy Approach for Entity and Relation Extraction</a>	<a href="#">GitHub</a>	<a href="#">Paper with Code</a>	2020
2	BERT-MRC	86.88	✓	<a href="#">A Unified MRC Framework for Named Entity Recognition</a>	<a href="#">GitHub</a>	<a href="#">Paper with Code</a>	2019

KU\_NLP  
(87.19)

## GENIA

Rank	Model	F1 ↑	Paper	Code	Result	Year
1	Biaffine-NER	80.5	<a href="#">Named Entity Recognition as Dependency Parsing</a>	<a href="#">GitHub</a>	<a href="#">Paper with Code</a>	2020
2	seq2seq+BERT+Flair	78.31	<a href="#">Neural Architectures for Nested NER through Linearization</a>	<a href="#">GitHub</a>	<a href="#">Paper with Code</a>	2019

KU\_NLP  
(83.73)





---

# 구문 분석 (Syntactic Parsing)

---

컴퓨터공학부 / 인공지능학과(대학원)

김 학 수

# Core Layers of NLU

단계	설명	예제: 나는 그 과자를 먹었다.
형태소 분석	문장을 형태소열로 분리하고 품사를 부착하는 단계	나/대명사+는/조사 그/대명사 과자/명사+를/조사 먹/동사+었/선어말어미+다/어미+./기호
구문 분석	문장의 문법적 적합성과 어절의 구문적 역할(주어, 목적어 등)을 찾는 단계	[SUBJ: 나는 [[MOD: 그 [OBJ: 과자를]] 먹었다]]
의미 분석	문장을 구성하는 술어와 논항들 사이의 의미적 적합성을 분석하는 단계	PREDICATE: 먹다 AGENT: 나/ANIMATE OBJECT: 그 과자/EATABLE
담화 분석	대화 문맥을 파악하여 상호참조를 해결하고 의도를 파악하는 단계	SPEECH ACT: STATEMENT PREDICATE: 먹다 AGENT: 홍길동/ANIMATE OBJECT: 꼬깔콘/EATABLE



# 구문 분석

---

- 구문 분석이란?
  - 문장 내 각 어절의 구문적 역할을 찾아내는 것
  - 예) 감기는 자주 걸리는 병이다.
    - '감기는'은 '병이다'의 주어
    - '자주'는 '걸리는'의 부사어
    - '걸리는'은 '병이다'의 수식어
- 활용 분야
  - Grammar checkers
  - Question answering
  - Information extraction
  - Machine translation



# 문법

---

- 문법이란?
  - 문장을 구성하는 규칙
  - 문장이 어떤 구성요소(constituency)로 이루어져 있으며, 어떤 순서(ordering)로 만들어지는 지에 대한 규칙
- 문법의 종류: 촘스키 계층
  - 정규 문법(RG; Regular Grammar) → 단어 분리기 수준
  - 문맥자유문법(CFG; Context-Free Grammar) → 구문 분석 수준
  - 문맥의존문법(CSG; Context Sensitive Grammar) → 의미 분석 수준
  - 무제한문법(Unrestricted Grammar) → 모든 분석기 수준



# Context?

- The notion of context in CFGs has nothing to do with the ordinary meaning of the word context in language
- All it really means is that the non-terminal on the left-hand side of a rule is out there all by itself (free of context)

$A \rightarrow B C$

CFG 표기법

Means that I can rewrite an  $A$  as  $a B$  followed by a  $C$  regardless of the context in which  $A$  is found

CFG



# Key Constituents (English)

---

- Noun phrases
- Verb phrases
- Prepositional phrases
- Sentences



# NPs

---

- NP → PRONOUN  
e.g., I came, you saw it, they conquered
- NP → PROPER-NOUN  
e.g., Los Angeles is west of Texas  
e.g., John Hennessy is the president of Stanford
- NP → DET NOUN  
e.g., The president
- NP → NOMINAL
- NOMONAL → NOUN NOUN  
e.g., A morning flight to Denver



# PPs

---

- PP → PREPOSITION NP

e.g., from LA

to the store

on Tuesday morning

with lunch





# Sentences

---

- Declaratives: A plane left  
 $S \rightarrow NP VP$
- Imperatives: Leave!  
 $S \rightarrow VP$
- Yes-No Questions: Did the plane leave?  
 $S \rightarrow AUX NP VP$
- WH Questions: When did the plane leave?  
 $S \rightarrow WH AUX NP VP$

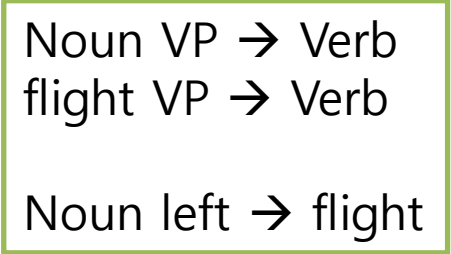


# CFG Example

- $S \rightarrow NP VP$
- $NP \rightarrow DET NOMINAL$
- $NOMINAL \rightarrow NOUN$
- $VP \rightarrow VERB$
- $DET \rightarrow a$
- $NOUN \rightarrow flight$
- $VERB \rightarrow left$



Context-sensitive  
grammar



Noun VP  $\rightarrow$  Verb  
flight VP  $\rightarrow$  Verb

Noun left  $\rightarrow$  flight



# 파싱(Parsing)

---

- 파싱(parsing)이란?
  - 구문 분석 과정
  - 문장과 문법을 입력으로 받아서 입력 문장을 커버하는 파스 트리를 만들어내는 과정

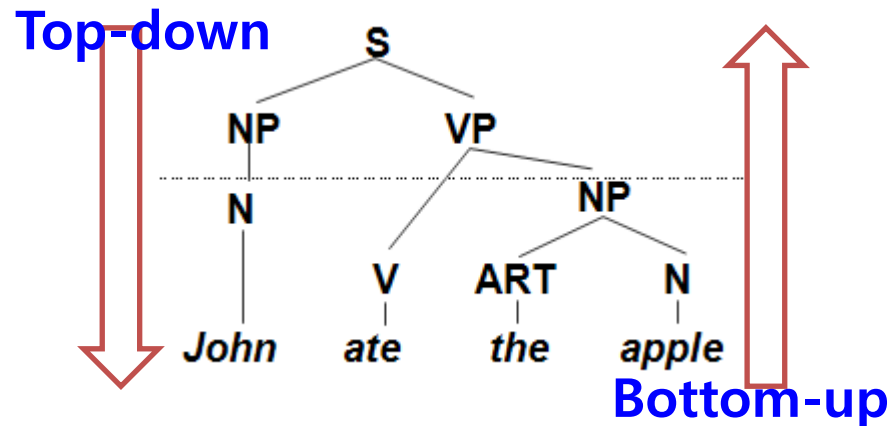


# Bottom-up parsing vs. Top-down parsing

- Bottom-up parsing
  - 입력 문장의 단어로부터 시작하여 rewrite 규칙을 backward로 적용하여 S 하나만 나타날 때까지 적용
- Top-down parsing
  - S부터 rewrite해서 주어진 sentence가 generate될 때까지 적용
  - 만일 모든 가능성을 다 적용해도 그 문장이 나타나지 않으면 parsing 실패

## Grammar

S → NP VP  
NP → ART N  
NP → N  
VP → V NP



# Bottom-up vs. Top-down

---

- Top-down
  - Only searches for trees that can be answers (i.e.  $S$ 's)
  - But also suggests trees that are not consistent with the words
- Bottom-up
  - Only forms trees consistent with the words
  - Suggest trees that make no sense globally



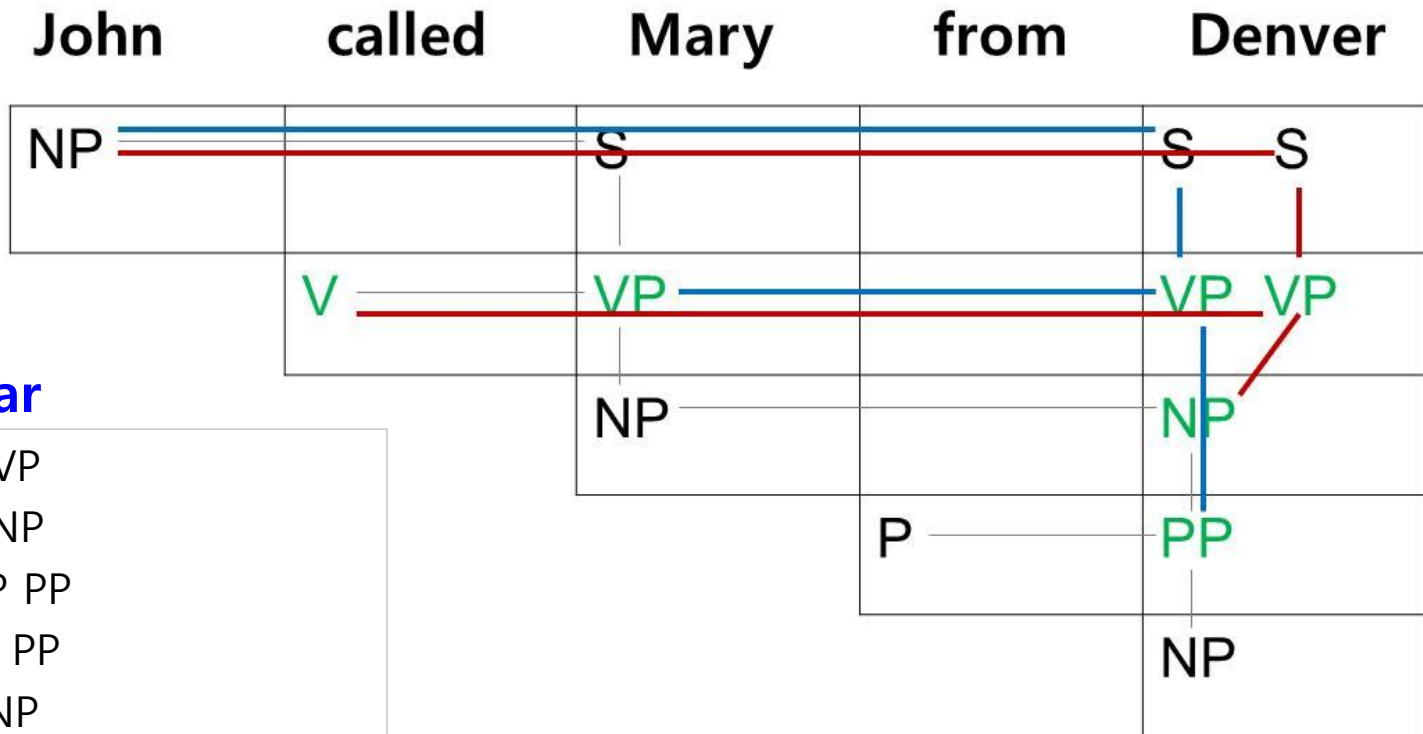
# CKY Algorithm

- 대표적인 Bottom-Up 파싱 알고리즘

```
function CKY-PARSE(words, grammar) returns table
  for  $j \leftarrow$  from 1 to LENGTH(words) do
     $table[j-1, j] \leftarrow \{A \mid A \rightarrow words[j] \in grammar\}$ 
    for  $i \leftarrow$  from  $j-2$  downto 0 do
      for  $k \leftarrow i+1$  to  $j-1$  do
         $table[i, j] \leftarrow table[i, j] \cup$ 
           $\{A \mid A \rightarrow BC \in grammar,$ 
             $B \in table[i, k],$ 
             $C \in table[k, j]\}$ 
```



# CKY Example



## Grammar

$S \rightarrow NP VP$

$VP \rightarrow V NP$

$NP \rightarrow NP PP$

$VP \rightarrow VP PP$

$PP \rightarrow P NP$

$NP \rightarrow \text{John, Mary, Denver}$

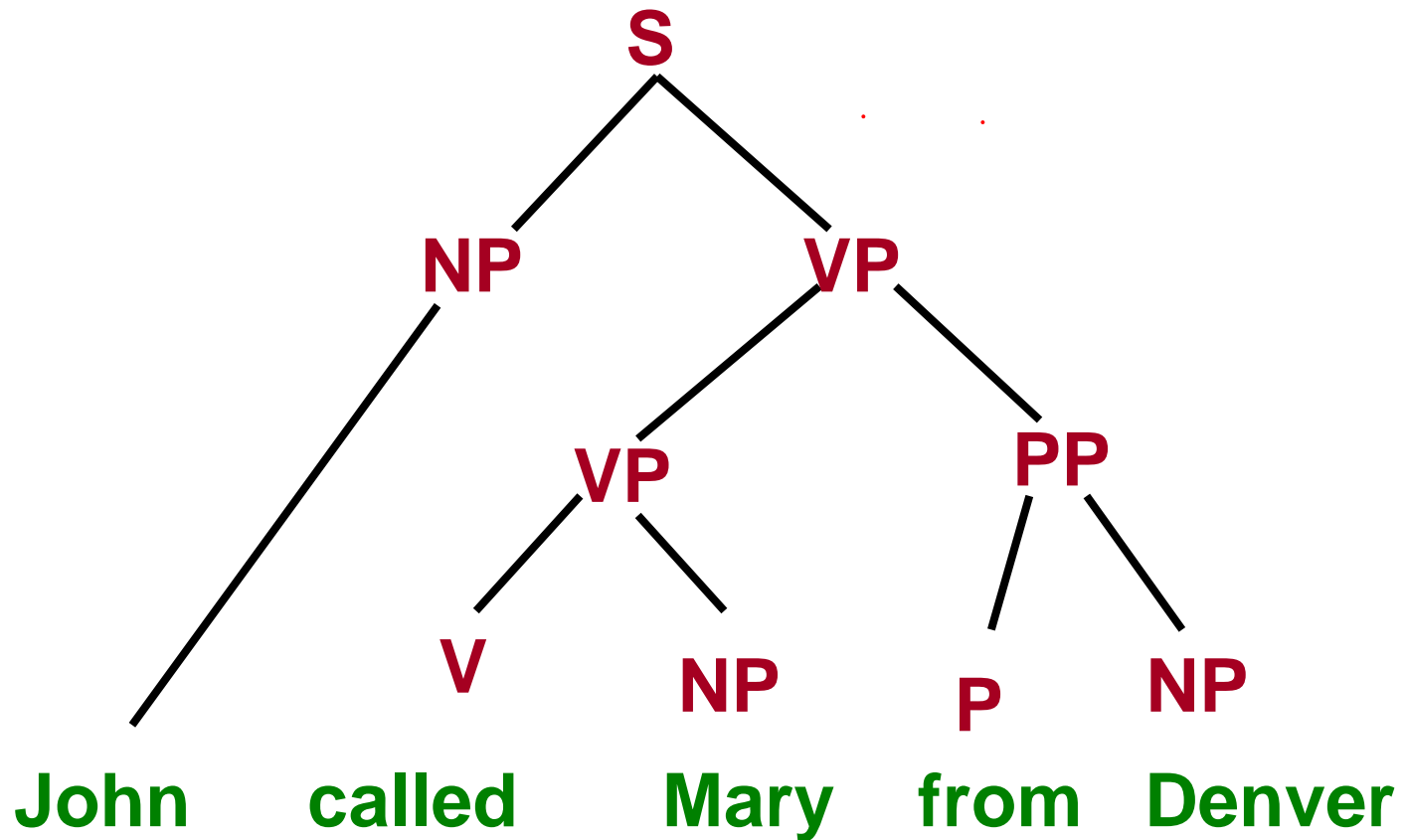
$V \rightarrow \text{called}$

$P \rightarrow \text{from}$



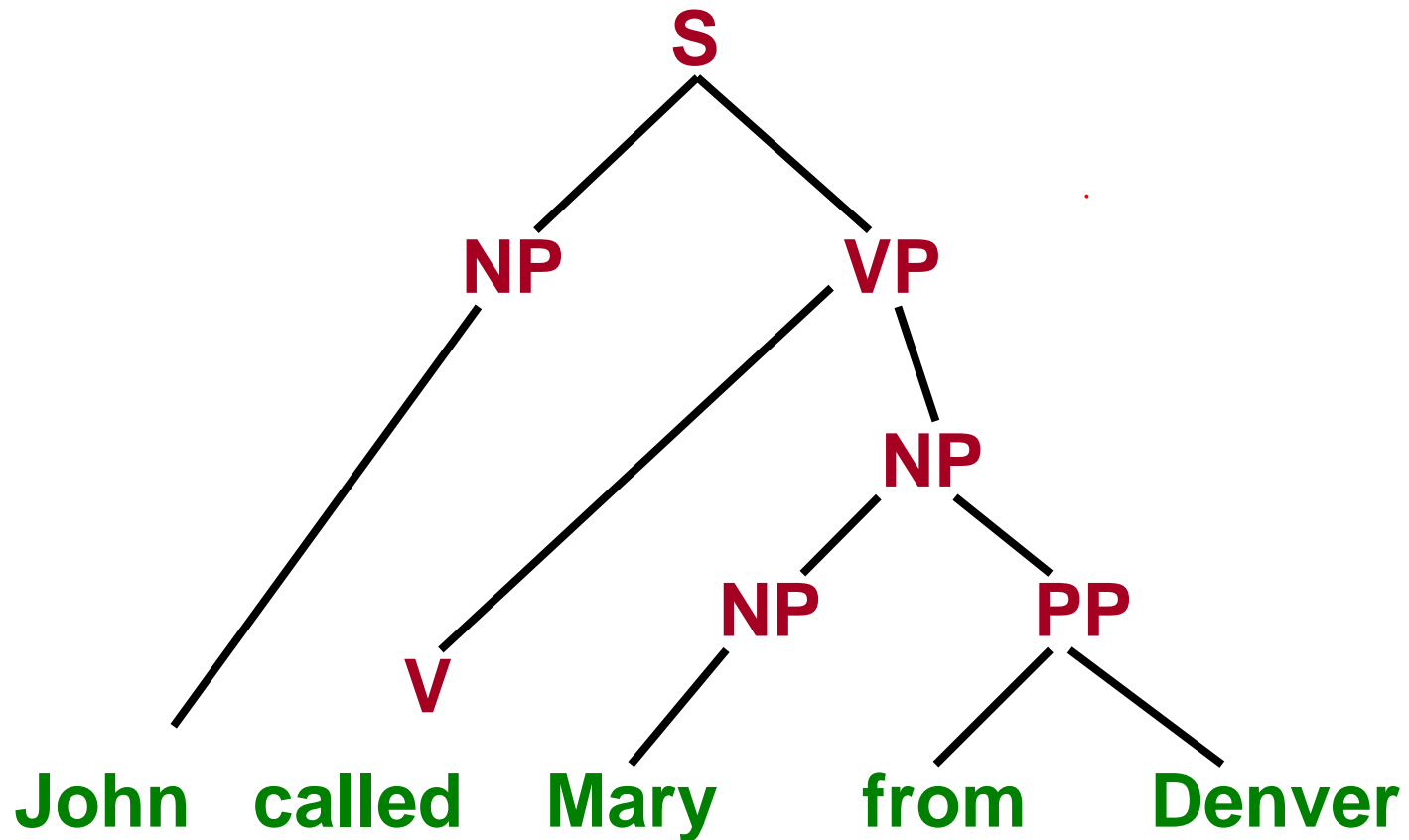
# CKY Example

---





# CKY Example



# Lots of Ambiguity

- Church and Patil (1982)
  - Number of parses for such sentences grows at rate of number of parenthesizations of arithmetic expressions
  - Which grow with Catalan numbers

$$C(n) = \frac{1}{n+1} \binom{2n}{n}$$

PPs	Parses
1	2
2	5
3	14
4	132
5	469
6	1430



# 해결책 → 확률

---

- Penn Treebank
  - UPen에서 구축한 구문 분석 말뭉치
  - 구문 분석 결과가 수동으로 부착된 대용량의 영문 말뭉치
- 간단한 애매성 해소법
  - 대용량의 구문 분석 말뭉치에서 적용된 **CFG** 문법의 확률 계산
  - 입력 문장을 파싱할 때 사용된 문법의 확률을 모두 곱함
  - 최대 확률을 갖는 파스 트리를 선택



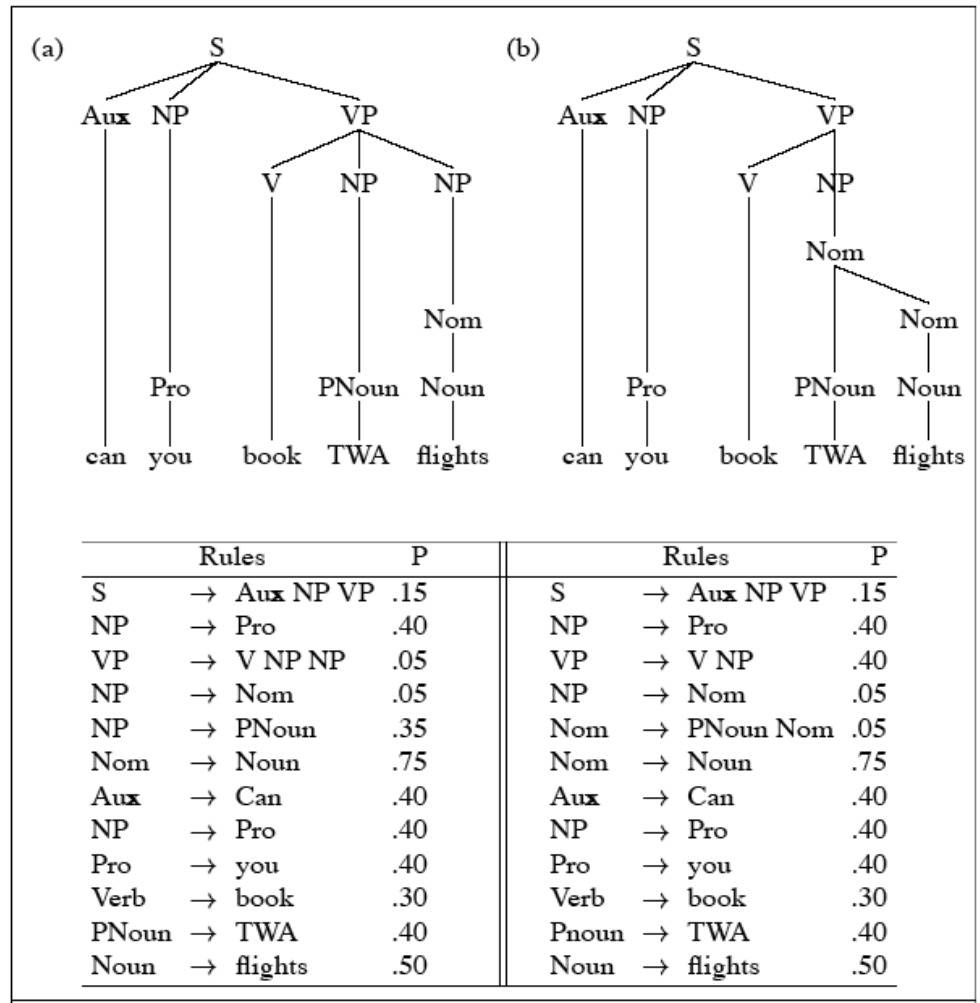
# 통계적 파싱(Statistical Parsing)

$$P(T, S) = \prod_{n \in T} p(r_n)$$

$$P(T, S) = P(T)P(S/T) = P(T); \text{ since } P(S/T)=1$$

$$\begin{aligned} P(T_i) &= .15 * .40 * .05 * .05 * .35 * .75 * .40 * .40 * .40 \\ &\quad * .30 * .40 * .50 \\ &= 1.5 \times 10^{-6} \end{aligned}$$

$$\begin{aligned} P(T_r) &= .15 * .40 * .40 * .05 * .05 * .75 * .40 * .40 * .40 \\ &\quad * .30 * .40 * .50 \\ &= 1.7 \times 10^{-6} \end{aligned}$$



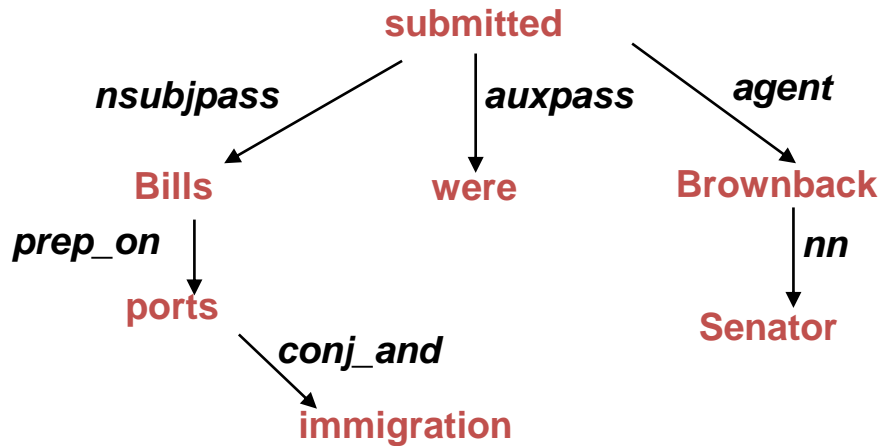
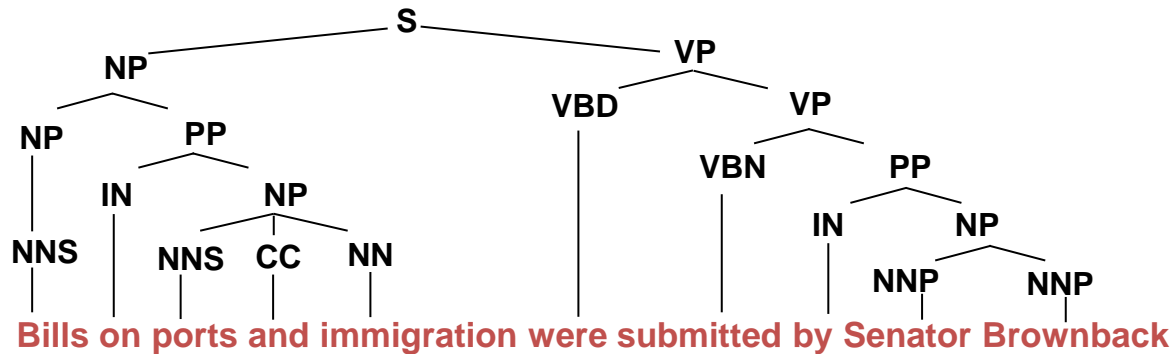
# 의존구조 분석 (Dependency Parsing)

---

- 구구조 분석(Phrase Structure Analysis)
  - 지금까지 배운 것과 같이 문장의 구성요소(constituency)인 구(Phrase)의 순서(ordering)을 기반으로 문법을 기술
  - 단어의 순서가 의미를 가지는 영어와 같은 언어에 적합
- 의존구조 분석(Dependency Structure Analysis)
  - 어절과 어절 사이의 관계를 문법으로 기술
  - 어순이 자유로운 한국어에 적합한 구문분석 방법



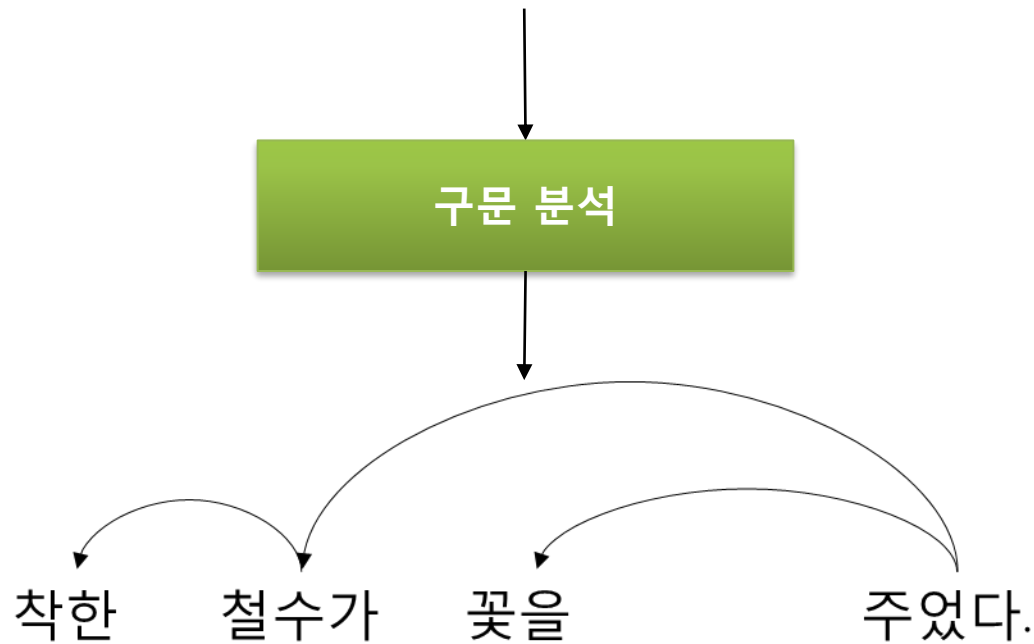
# Phrase Structure vs Dependency Structure



# 한국어 의존구조 분석

- 입력 : 형태소 분석 결과
- 출력 : 의존 구문 트리

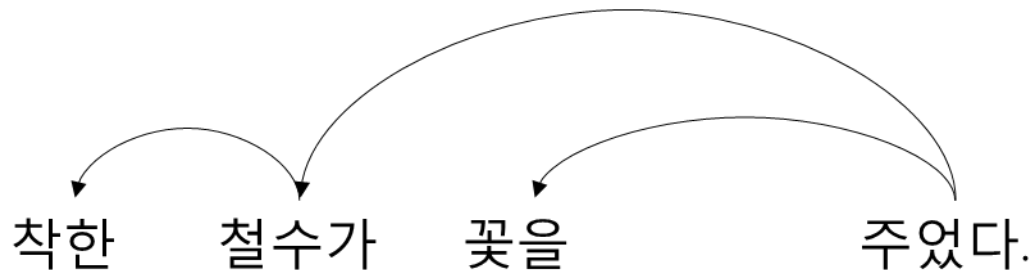
착하/VA+ㄴ/ETM 철수/NNG+가/JKS 꽃/NNG+을/JKO 주/VV+었/EP+다/EF+./SF



# 의존소 vs 지배소

- 지배소와 의존소
  - 지배소(Head) : 의미의 중심이 되는 요소
  - 의존소(Dependent) : 지배소가 갖는 의미를 보완해주는 요소

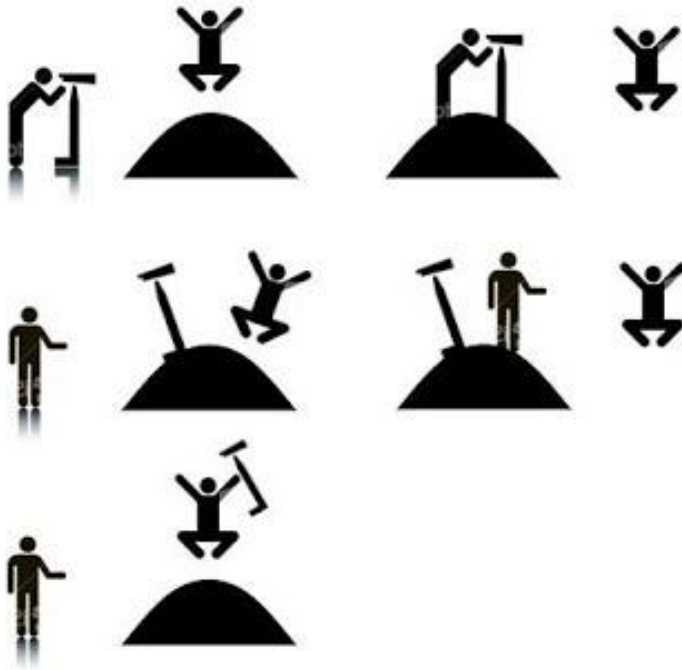
예) 주었다.(H) → 꽃을(D)  
철수가(H) → 착한(D)





# 애매성 해결

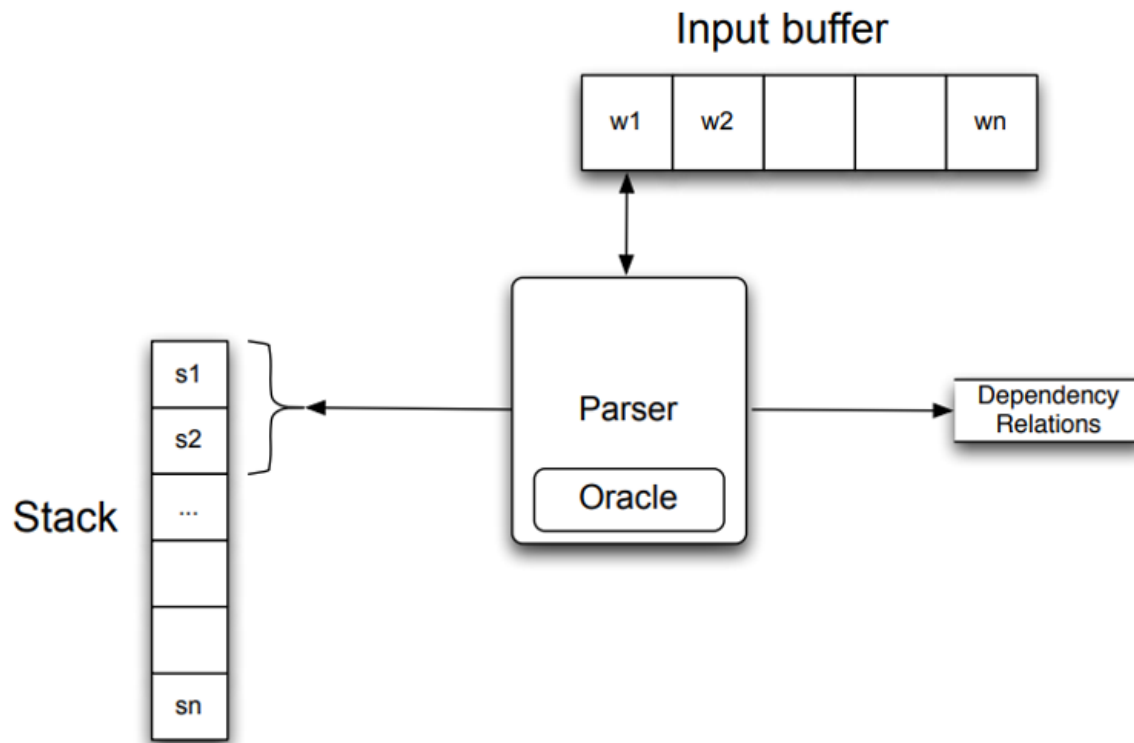
- 구문 분석에서의 애매성
  - “I saw the man on the hill with a telescope”



- 기계 학습을 이용하여 애매성을 해결

# Transition-Based Model

- Deterministic Shift-Reduce Parsing:  $O(n)$



<https://web.stanford.edu/~jurafsky/slp3/14.pdf>



# Shift Reduce Parsing 알고리즘

```
forward_parsing()
{
    Queue in  = [e2, ..... , en];
    Stack out = [e1];

    while (not out.empty()) {
        dep_cand = out.top(); // ei
        head_cand = in.first(); // ej

        x = get_context_features(dep_cand, head_cand);
        y = estimate_action(model, x);

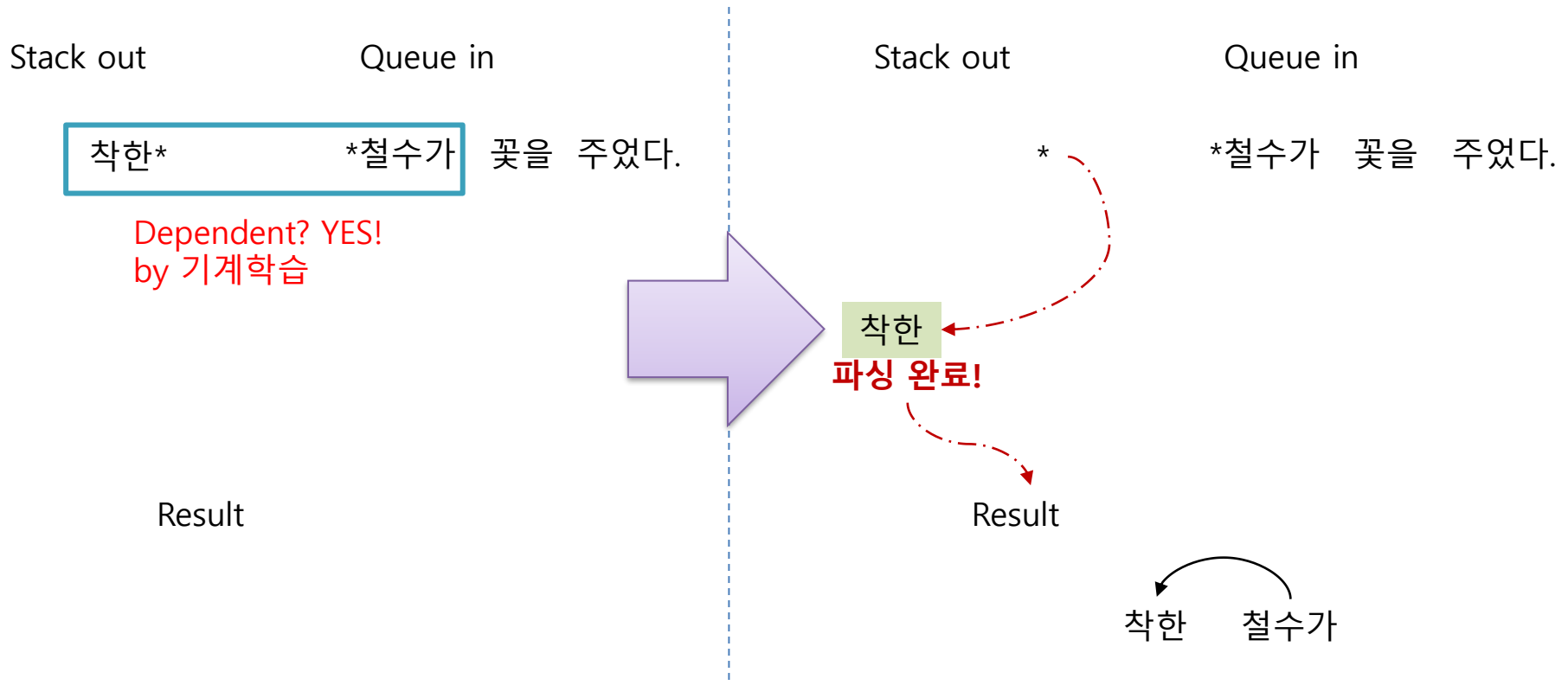
        if (y == DEPEND) {
            set_head(dep_cand, head_cand);
            out.pop();
        }
        else if (y == SHIFT) {
            out.push(in.pop());
        }
    }
}
```

기계학습을 이용한 의존  
관계 예측



# Shift Reduce Parsing 예제

입력 문장: 착한 철수가 꽃을 주었다.



# Shift Reduce Parsing 예제

입력 문장: 착한 철수가 꽃을 주었다.

Stack out

Queue in

\* < — . — \***철수가** 꽃을 주었다.

스택으로 이동

Result

착한      철수가



# Shift Reduce Parsing 예제

입력 문장: 착한 철수가 꽃을 주었다.

Stack out

Queue in

철수가\*

\*꽃을

주었다.

Dependent? NO!  
by 기계학습

Stack out

Queue in

철수가\*

← · — · — \*꽃을

주었다.

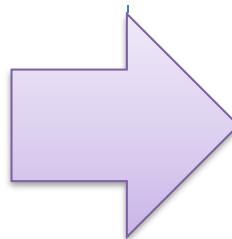
스택으로 이동

Result

착한    철수가

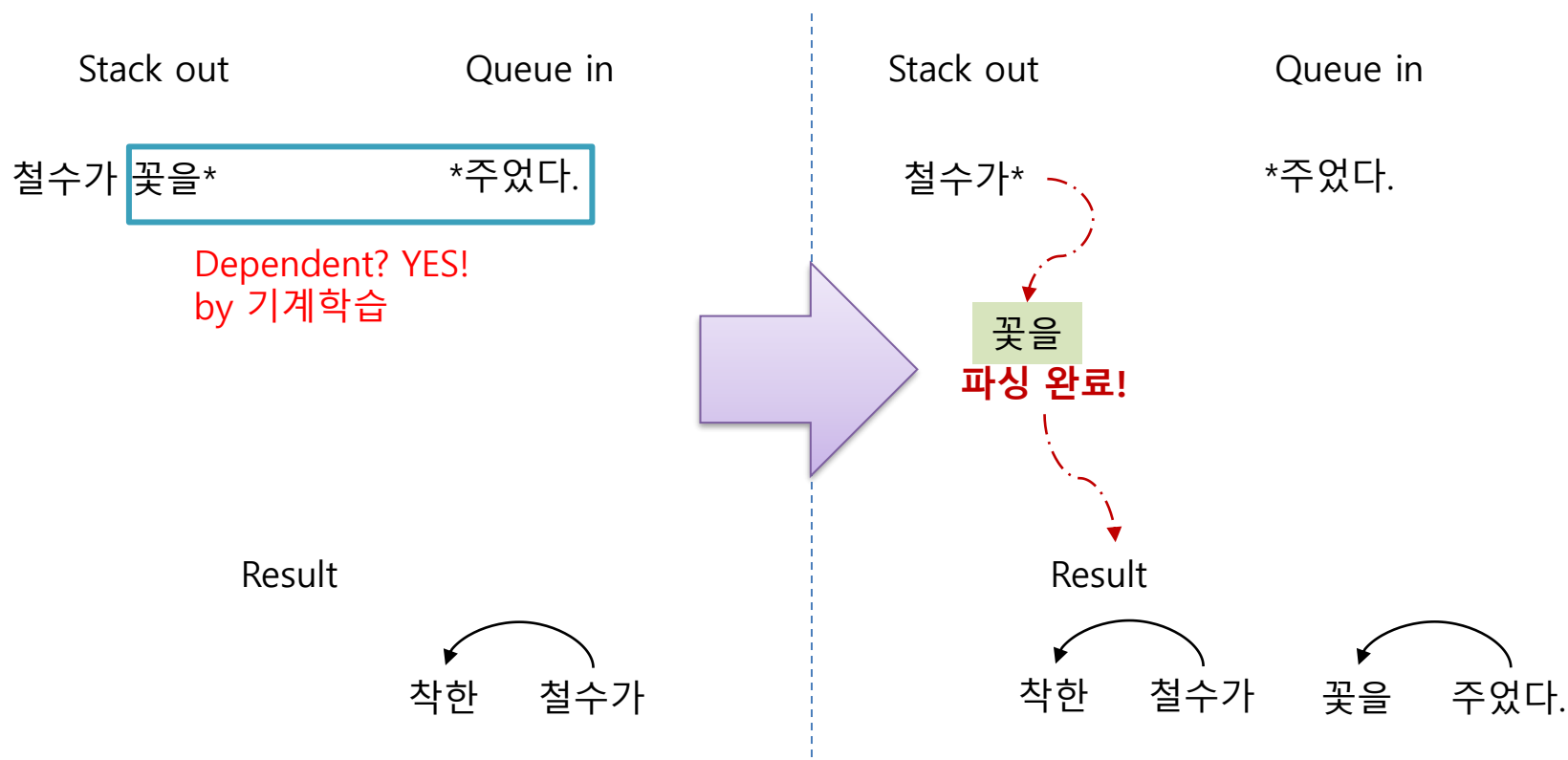
Result

착한    철수가



# Shift Reduce Parsing 예제

입력 문장: 착한 철수가 꽃을 주었다.



# Shift Reduce Parsing 예제

입력 문장: 착한 철수가 꽃을 주었다.

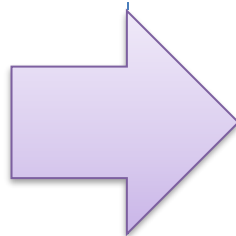
Stack out

Queue in

철수가\*

\*주었다.

Dependent? YES!  
by 기계학습



Stack out

Queue in

\*

\*

철수가

주었다.

파싱 완료!

Result

Result

착한

철수가

꽃을

주었다.

착한

철수가

꽃을

주었다.

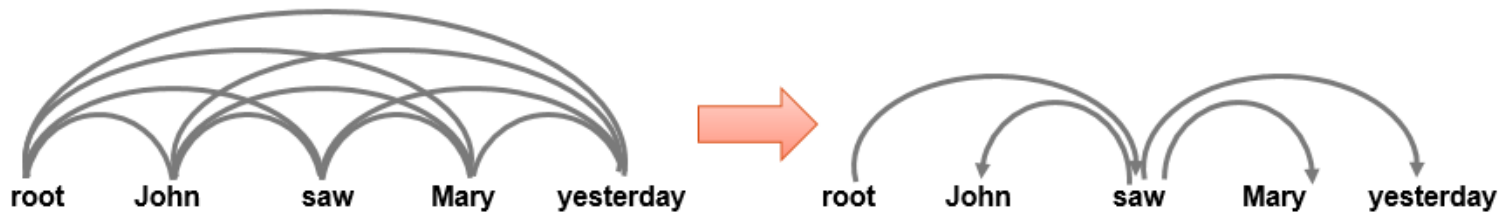




# Graph-Based Parsing

- How to find  $Y^*$

$$Y^* = \operatorname{argmax}_{Y \in \Phi(X)} \operatorname{score}(X, Y)$$



- 1st Order  $O(n^3)$ , Eisner's Algorithm**

- Dependencies are independent from each other
- Decompose  $\rightarrow$  arc-factorization

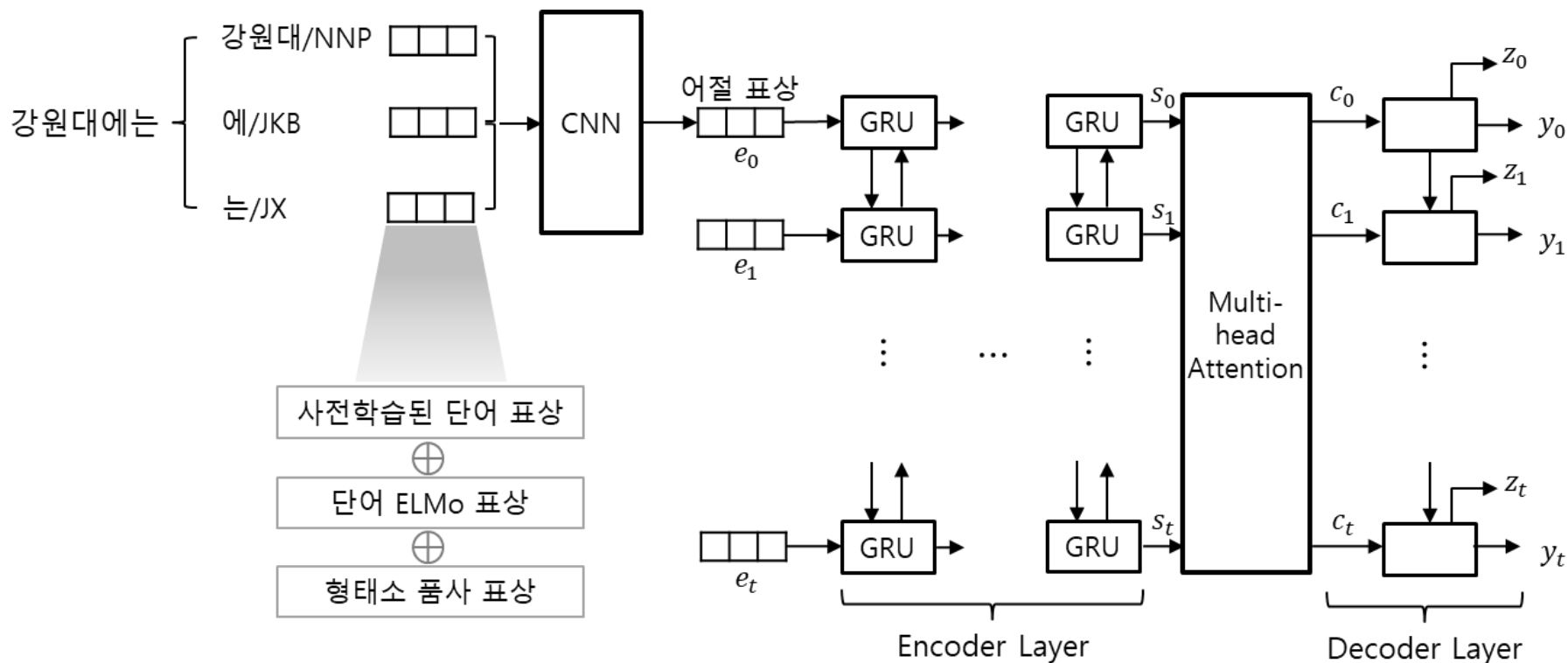


$$\operatorname{Score}(X, Y) = \sum_{(h,d) \in Y} \operatorname{score}(X, h, d)$$

\* Figure by kjlee in CNU

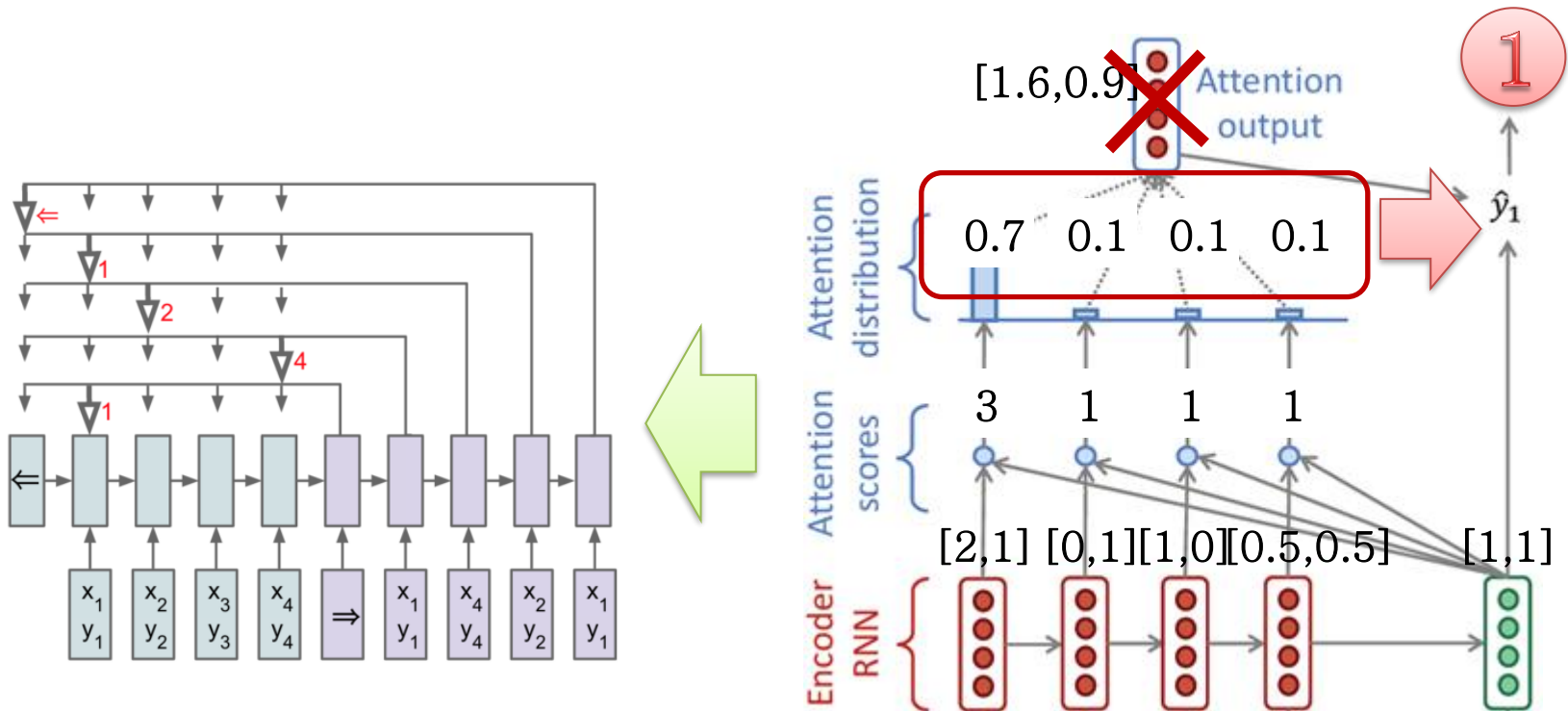


# 포인터 네트워크를 이용한 의존구조 분석



# 포인터 네트워크

- Seq2Seq+Attention 모델 → 인덱스를 리턴하는 포인터 네트워크



# 실험 및 평가

---

- 실험 환경
  - 20GB 뉴스기사 데이터를 사용해 형태소 임베딩 사전 학습
  - 사용한 데이터: 세종 말뭉치
    - 학습 데이터 48,458 문장
    - 개발 데이터 5,384 문장
    - 평가 데이터 5,817문장



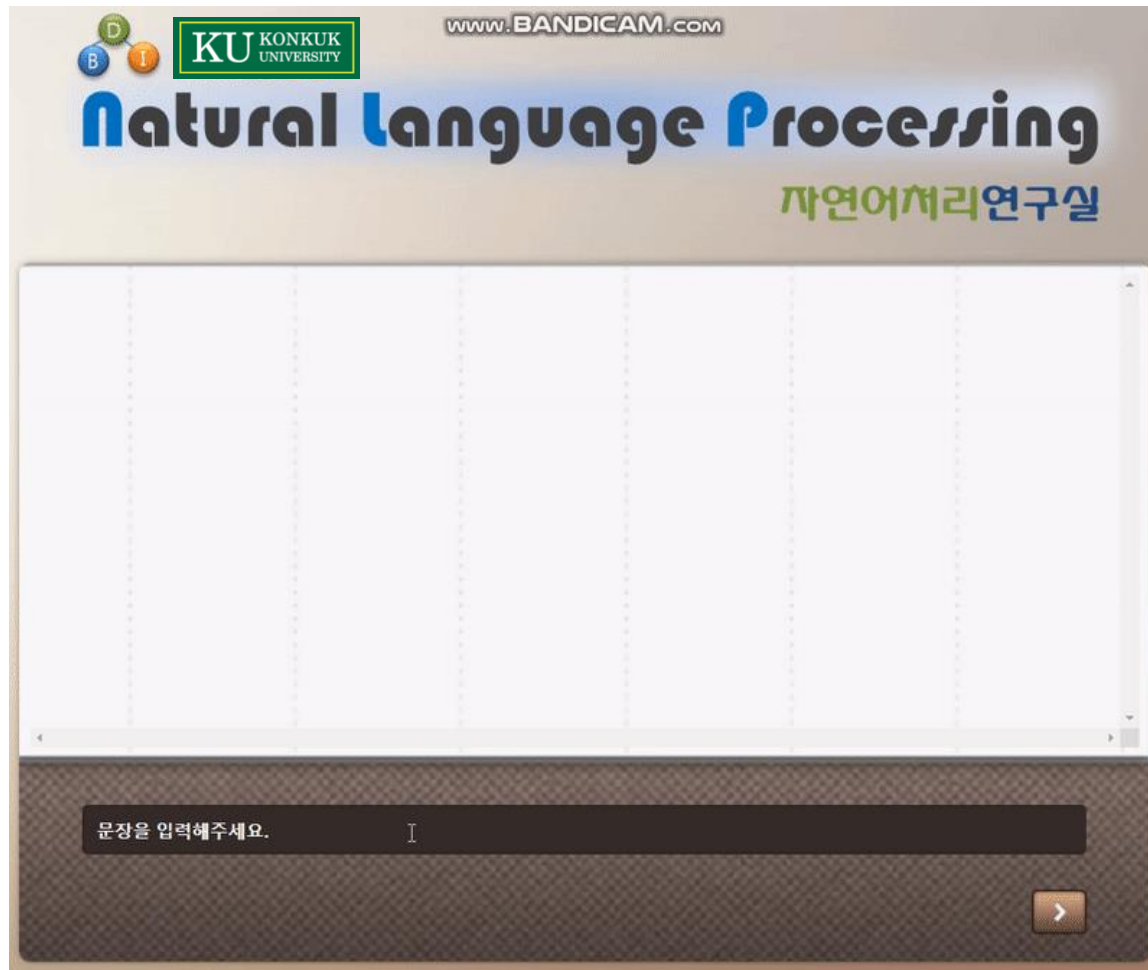
# 실험 및 평가

- 동일 데이터를 사용한 타연구 성능 비교

	UAS	LAS
이창기 외, 2014 [1], FNN 기반 모델	0.9037	0.8817
안재현 외, 2017 [2], Pointer network	0.9069	0.8750
나승훈 외, 2016 [3], Stack-LSTM	0.9083	0.8849
나승훈 외, 2017 [4], Biaffine Attention	0.9178	0.8976
박천음 외, 2017 [5], Pointer network	0.9179	0.8948
<b>제안 모델</b>	<b>0.9285</b>	<b>0.9065</b>



# 의존 구조 분석 시연 영상



# 질의응답

---

Q&A

Homepage: <http://nlp.konkuk.ac.kr>  
E-mail: [nlpdrkim@konkuk.ac.kr](mailto:nlpdrkim@konkuk.ac.kr)

