CS224n 5강 발표

2024.10.29

발표자 : 나상현

Sentence Structure (문장 구조)

문장 구조 파악의 중요성:
문장의 구조를 파악해야 문장의 의미를 정확히 이해할 수 있음

두 가지 방법:

- **Phrase-Structure Grammar (구문 구조 문법)**: → (=CFG)
  문장을 **구 단위**로 나누어 구조적 관계 파악

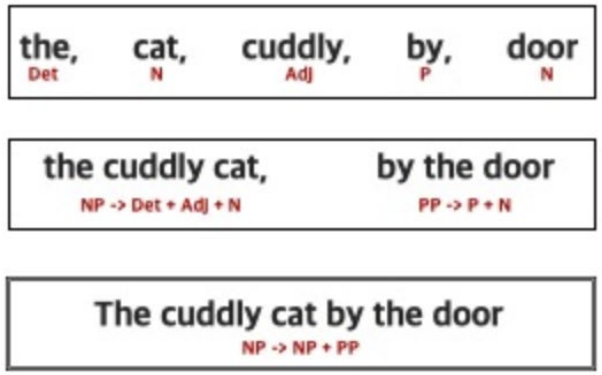
- **Dependency Structure Grammar (의존 구조 문법)**
  단어 간의 의존 관계로 문장의 의미를 분석

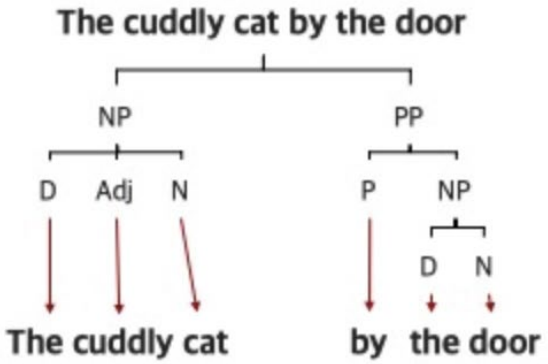
응용:
컴퓨터가 문장을 이해하고 자연어 처리를 수행하는 기반 기술
NLP에서 parsing은 구문 트리를 구성해 문장의 논리적 흐름을 파악함으로써 기계 학습 모델의 성능을 높임

Phrase-Structure Grammar (구문 구조 문법)

정의:
문장을 구(Phrase) 로 나누고, 이들을 중첩(Nesting)하여 문장의 계층적 구조를 표현
Context-Free Grammar (CFG) 기반으로 구문을 트리 구조로 표현

구성 방식:
문장의 구성 요소를 점차 큰 구로 묶어가며 최종적으로 전체 문장을 구조적으로 표현
예를 들어 **"The cuddly cat by the door"**는 **명사구(NP)**와 **전치사구(PP)**로 구분되며, 각 단어의 문법적 기능을 파악 가능





Phrase-Structure Grammar (구문 구조 문법)

Starting unit: words are given a category (part of speech = pos)

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

Words combine into phrases with categories

the cuddly cat, by the door
NP → Det Adj N PP → P NP

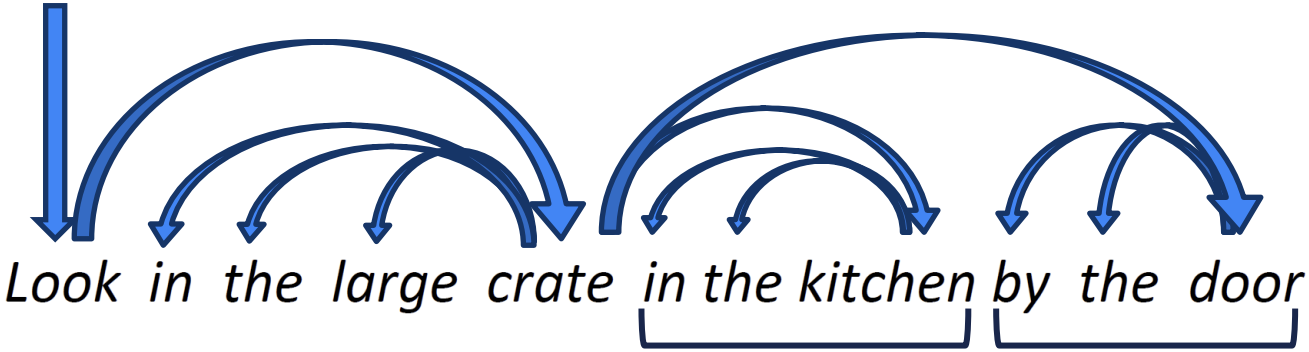
Phrases can combine into bigger phrases recursively

the cuddly cat by the door
NP → NP PP

Dependency Structure Grammar (의존 구조 문법)

정의:
각 단어가 다른 단어에 의존적으로 관계를 형성하는 방식
의존 관계는 화살표로 시각화되며, 수식을 받는 단어는 Head, 수식하는 단어는 Dependent로 구분.

구성 방식:
예를 들어 **"Look in the crate"**에서 **'Look'**이 문장의 중심(Head), **'in the crate'**는 수식하는 **Dependent**로 표현됨
화살표를 통해 각 단어가 어떤 단어에 의존적인지 시각화하고, 그 관계를 Label로 표현하여 문장 의미를 파악



Phrase Attachment Ambiguity(구문 부착 모호성)

Scientists count whales from space

✓

Scientists count whales from space

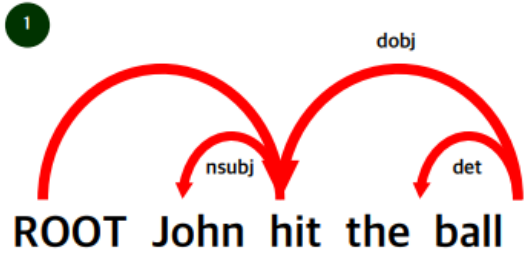
Phrase Attachment Ambiguity(구문 부착 모호성)

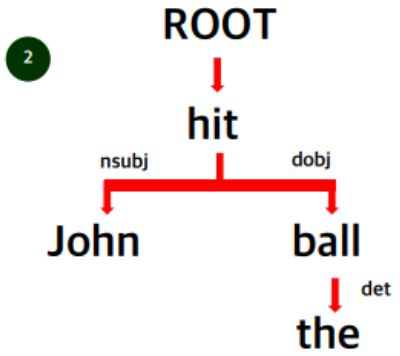
Dependency Grammar

Sequence: 화살표를 사용해 수식하는 단어(head)에서 수식받는 단어(dependent)로 방향성 표시
→ 단어 간의 관계를 화살표로 나타냄

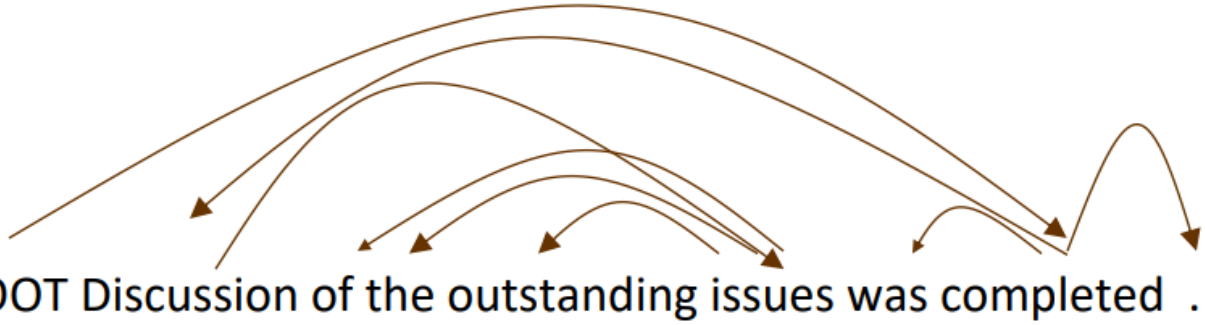
Tree: 수식받는 단어들이 트리 구조로 표현되며, 각 단어 사이의 의존 관계를 나타냄
→ 단어 간의 관계를 트리 구조로 나타냄

화살표의 방향은 항상 수식을 받는 단어(head)에서 수식하는 단어(dependent)로 향함





Dependency Grammar

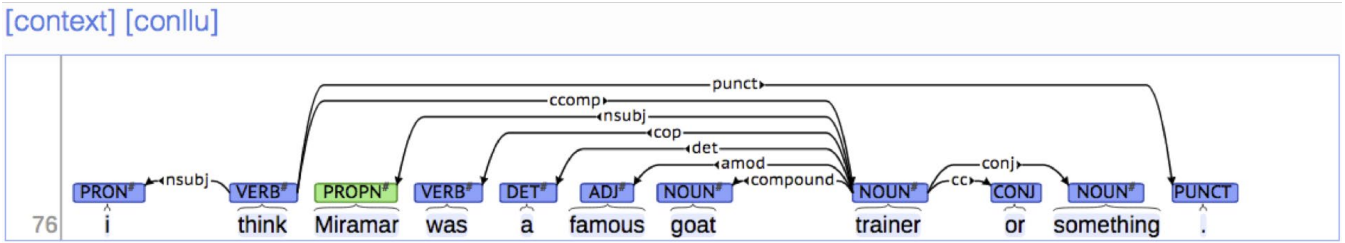


기본 규칙
ROOT 노드: 문장의 최상단에 위치하여 모든 단어가 연결되도록 설정
- 어떤 단어도 수식을 받지 않을 경우, ROOT와 연결되어야 함
- 이를 통해 모든 단어가 의존 관계를 가질 수 있도록 보장

특징
Bilexical Affinities: 두 단어 간의 의미적 친밀도 (e.g., "discussion"과 "issues")
Dependency Distance: 대개 가까운 위치에 있는 단어들 간에 관계가 형성됨
Intervening Material: 구두점 등이 단어 간 관계에 방해되지 않음
Valency of Heads: 한 단어가 몇 개의 dependent를 가질지 예측 가능

Treebank와 Dependency Parsing 데이터셋

Treebank:
사람이 직접 문장의 Dependency 관계를 설정한 데이터셋
다양한 언어의 문법적 구조를 수집하여 의존 구문 분석 학습에 사용



Universal Dependencies 프로젝트:
다국어를 아우르는 보편적인 Dependency 구조를 제공하여 NLP 연구에 활용
영어, 중국어, 프랑스어뿐만 아니라 다양한 언어를 포함하여 보편적 문법 패턴을 학습할 수 있음

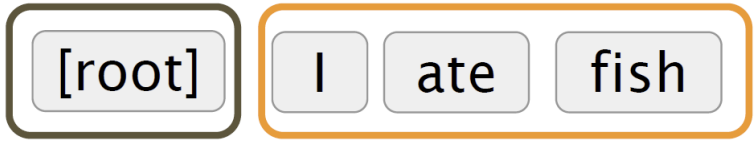
Dependency Parsing의 주요 기법

Dependency Parsing Method:
Dynamic Programming: 문장을 부분으로 나누어 트리 구축
Graph Algorithm: 가능한 모든 의존 관계를 고려해 최적의 트리 선택
Constraint Satisfaction: 문법적 제한 조건을 만족하는 구조를 구성
Transition-Based Parsing: 순차적으로 의존 관계를 설정하여 점진적 트리 구성

Transition-Based Parsing:
가장 많이 사용되는 방식으로, 빠르고 효율적으로 트리를 생성
단순한 규칙을 바탕으로 단계별로 단어 간 의존 관계를 결정하여 parsing을 완성하는 방법

Greedy Transition-Based Parsing

Start



Shift

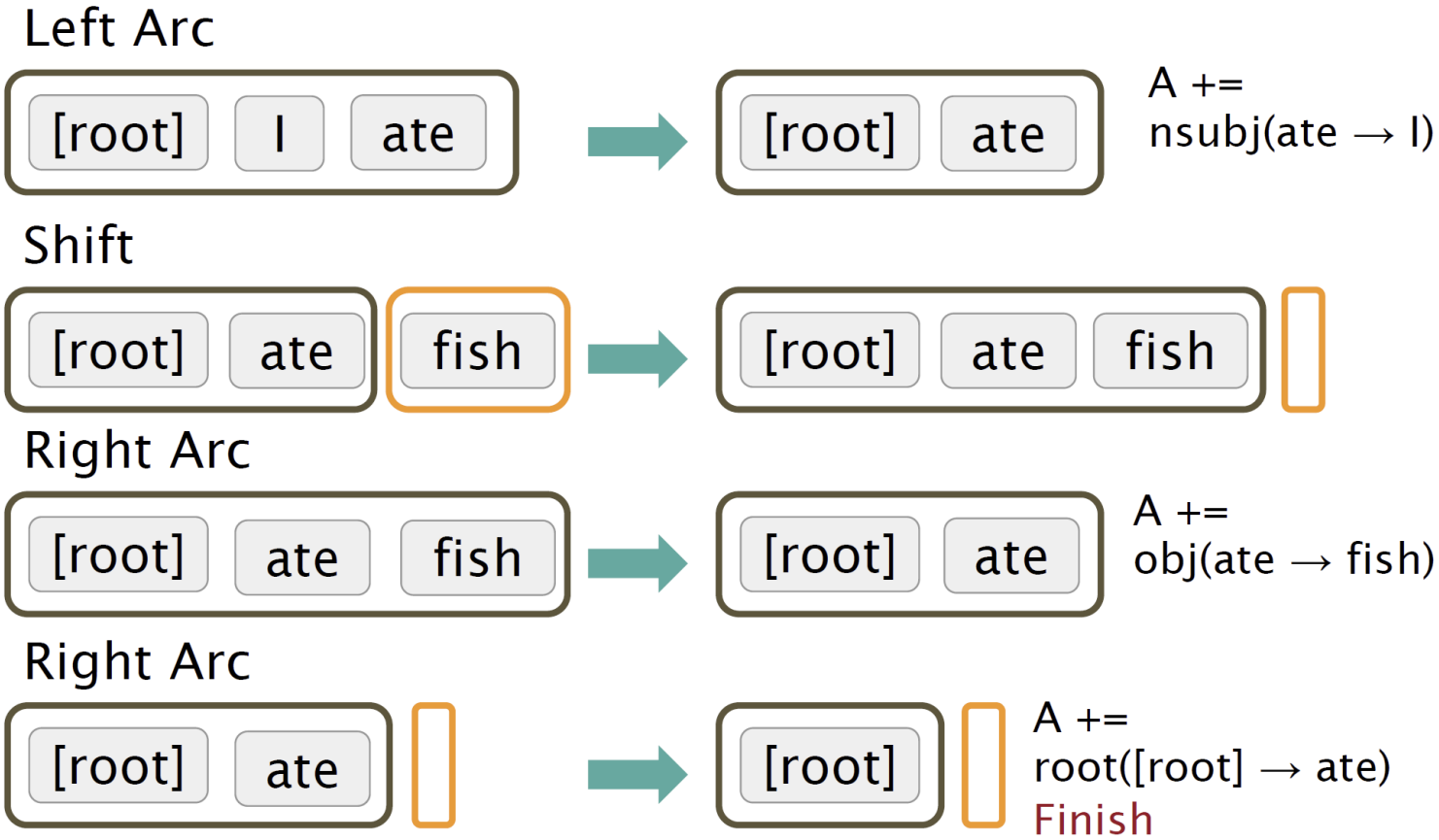
Shift

Start: $\sigma = [ROOT], \beta = w_1, \ldots, w_n, A = \emptyset$

1. Shift $\sigma, w_i|\beta, A \rightarrow \sigma|w_i, \beta, A$
2. Left-Arc$_r$ $\sigma|w_i|w_j, \beta, A \rightarrow \sigma|w_j, \beta, A \cup \{r(w_j, w_i)\}$
3. Right-Arc$_r$ $\sigma|w_i|w_j, \beta, A \rightarrow \sigma|w_i, \beta, A \cup \{r(w_i, w_j)\}$

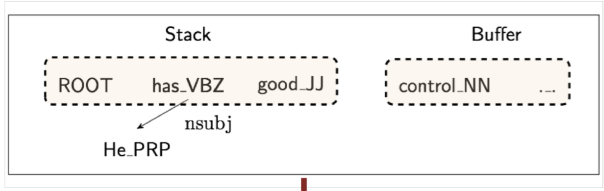
Finish: $\beta = \emptyset$

Greedy Transition-Based Parsing

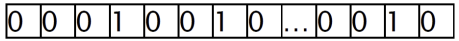


MaltParser와 C&M Parser 비교

MaltParser:
다음 액션을 softmax classifier로 예측하여, Shift, Left/Right Arc를 선택
Linear Time Parsing 제공하지만, 최상의 성능은 아님



binary, sparse
dim =10^6 ~ 10^7



Feature templates: usually a combination of 1 ~ 3 elements from the configuration.

Indicator features

$$s1.w = \text{good} \wedge s1.t = \text{JJ}$$
$$s2.w = \text{has} \wedge s2.t = \text{VBZ} \wedge s1.w = \text{good}$$
$$lc(s_2).t = \text{PRP} \wedge s_2.t = \text{VBZ} \wedge s_1.t = \text{JJ}$$
$$lc(s_2).w = \text{He} \wedge lc(s_2).l = \text{nsubj} \wedge s_2.w = \text{has}$$

C&M Parser (Chen & Manning, 2014):
MaltParser와 유사하게 동작하되, POS, Dependency Label 등도 활용하여 정확도 및 속도 개선
Neural Dependency Parsing 도입 이후로 빠른 속도와 높은 성능을 제공함

Neural Dependency Parser (신경망 기반 의존 구문 분석기)

Neural Dependency Parser의 필요성:
기존 방식의 단점을 보완하고, 문장의 의존 관계를 효율적으로 학습
Dense Feature Representation을 통해 희소성을 줄이고 계산 효율성을 향상

구성 요소:
Word Embedding, POS Tag, Dependency Label을 사용하여 단어 간 상호 관계 학습
Softmax를 사용하여 다음 액션(Shift, Left-Arc, Right-Arc) 예측

MaltParser와 비교:
MaltParser는 Linear Time Parsing을 제공하나, 성능이 다소 떨어짐
Neural Dependency Parser는 성능과 속도를 동시에 확보할 수 있도록 개선

Parser	UAS	LAS	sent. / s
MaltParser	89.8	87.2	469
MSTParser	91.4	88.1	10
TurboParser	**92.3**	89.6	8
C & M 2014	92.0	**89.7**	**654**

Dependency Parsing 성능 평가

Parser	UAS	LAS	sent. / s
MaltParser	89.8	87.2	469
MSTParser	91.4	88.1	10
TurboParser	**92.3**	89.6	8
C & M 2014	92.0	**89.7**	**654**

평가 지표:
Unlabeled Attachment Score (UAS): 의존 관계(Arc)가 올바른지 평가 (Label 미포함)
Labeled Attachment Score (LAS): 의존 관계와 Label이 모두 올바른지 평가

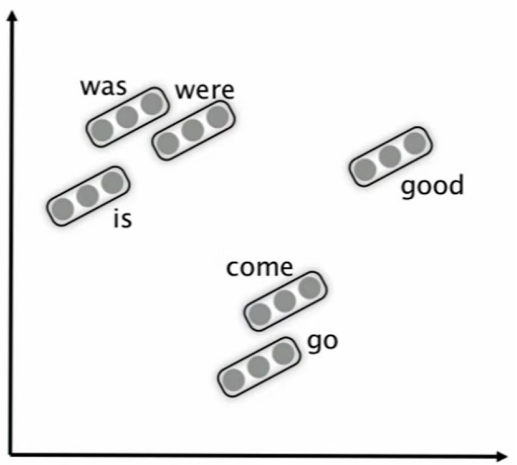
평가 예시:
English Penn Treebank Dataset을 활용한 평가 결과
MaltParser, Graph-based Parser, Neural Dependency Parser 성능 비교
Neural Parser가 속도와 성능 면에서 균형을 이루어 NLP 모델로써 탁월한 선택임을 보여줌

Distributed Representations

단어를 d-차원 벡터로 표현 (Word Embedding)
→ 유사한 단어들이 가까운 위치에 배치됨

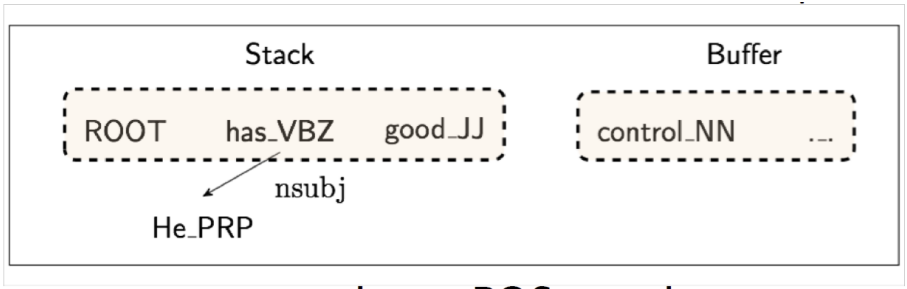
품사 태그와 의존 관계 라벨도 d-차원 벡터로 표현
→ 예: 복수 명사(NNS)와 단수 명사(NN)가 비슷하게 배치됨

수식어(num)와 형용사 수식어(amod) 등 유사한 라벨끼리 가까이 위치



Extracting Token Representations

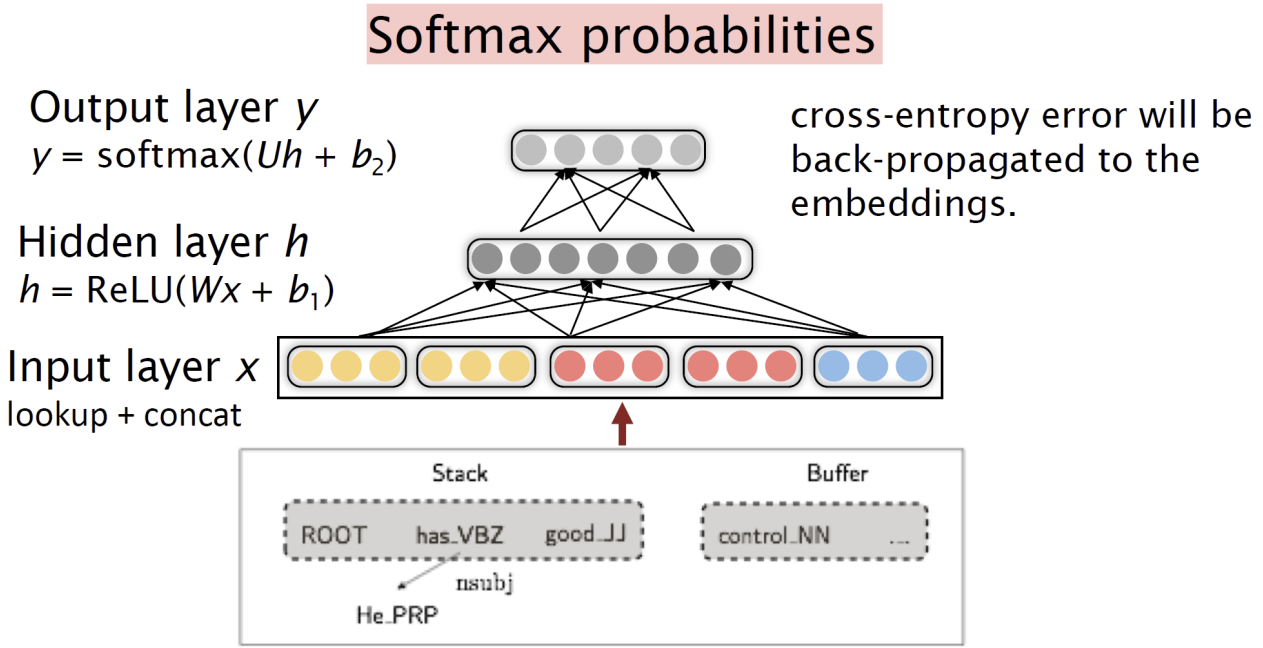
스택과 버퍼 위치에서 토큰을 추출함
이때 각 토큰은 단어, 품사, 의존 관계로 구성됨
→ 벡터 임베딩을 통해 이 정보를 결합하여 하나의 입력 벡터 생성



		word		POS		dep.
s_1		good		JJ		∅
s_2		has		VBZ		∅
b_1		control		NN		∅
$lc(s_1)$	→	∅	+	∅	+	∅
$rc(s_1)$		∅		∅		∅
$lc(s_2)$		He		PRP		nsubj
$rc(s_2)$		∅		∅		∅

Neural Network Model Architecture

입력층: 벡터 임베딩 결합
은닉층: ReLU 활성화 함수를 사용
출력층: Softmax로 Shift, Left Arc, Right Arc 중 하나 선택
=> Cross-Entropy 오차 역전파로 학습



Evaluation and Recent Advances

Chen & Manning (2014): 신경망을 활용해 높은 성능과 속도를 구현한 최초의 의존 구문 분석기
Google 연구팀의 추가 개선: Beam Search 도입으로 정확도 95%에 도달
새로운 신경망 기반 파서들이 기존의 오류율을 반으로 줄이며 발전

Method	UAS	LAS (PTB WSJ SD 3.3)
Chen & Manning 2014	92.0	89.7
Weiss et al. 2015	93.99	92.05
Andor et al. 2016	94.61	92.79

감사합니다