의존구문분석
(Dependency Parsing)

남궁영
한국해양대학교 컴퓨터공학과
young_ng@kmou.ac.kr

2019. 03. 29.

국립 한국해양대학교
KOREA MARITIME AND OCEAN UNIVERSITY

Content

❖ Dependency Parsing에 대해서는 약 3번에 걸쳐 발표 예정

→ 매 주 공부하고 실험한 내용을 정리해서 발표하고
그 결과를 심층학습 시간에 통합하여 발표 예정 (4월 17일)

❖ Dependency Grammar and Dependency Structure

- Dependency Parsing
- Transition-Based Dependency Parsing
- Neural Dependency Parsing
  - Feature Selection
  - FNN Model

Lecture Plan

- D. Jurafsky, J. H. Martin, Dependency Parsing, 「Speech and Language Processing(3rd ed.)」, 2018
- C. Manning, CS224n: Natural Language Processing with Deep Learning(winter 2017), Stanford Univ.

❖ Syntactic Structure: Consistency and Dependency

❖ Dependency Grammar

❖ Transition-based dependency parsing

❖ Neural dependency parsing

I Syntactic Structure

❖ Two views of linguistic structure:
constituency = phrase structure grammar = context-free grammars (CFGs)

- Phrase structure organizes words into nested constituents

I Syntactic Structure

Constituency structures

= phrase structure grammar
= context-free grammars(CFGs)

- Phrase structure organizes words into nested constituents

S → NP VP

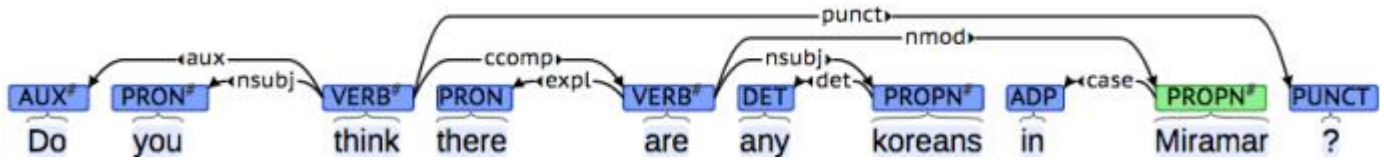
VP → V NP

NP → Det N

…

Dependency structures

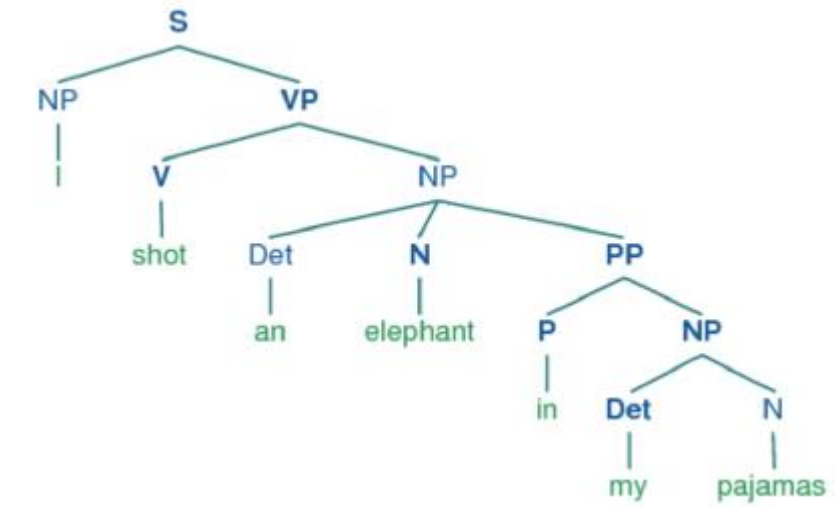
- Dependency structure shows which words depend on (modify of are arguments of) which other words.



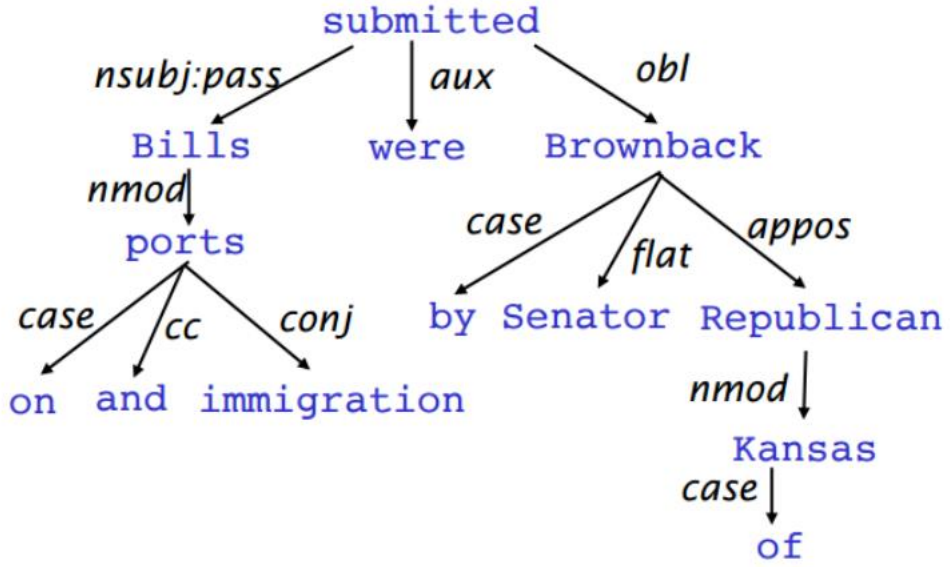
| https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1174/lecture_notes/cs224n-2017-notes4.pdf

I Syntactic Structure

Constituency structures

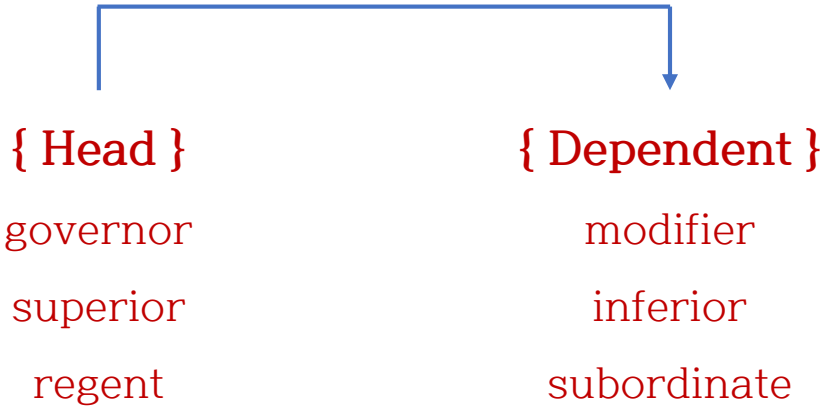


Dependency structures



Ⅱ Dependency Grammar and Dependency Structure

❖ Dependency syntax postulates that syntactic structure consists of relations between lexical items, normally binary asymmetric relations (“arrows”) called dependencies



{ Head }

governor

superior

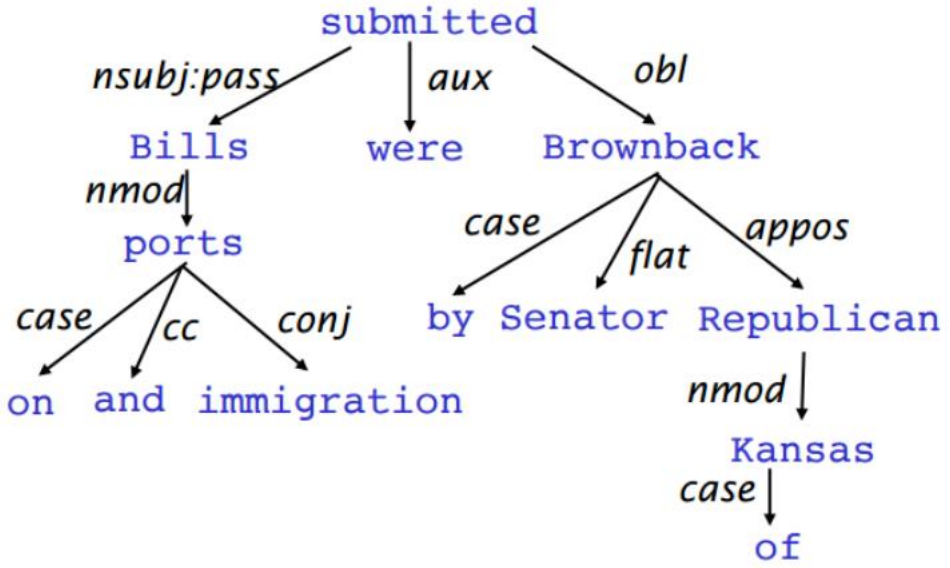
regent

{ Dependent }

modifier

inferior

subordinate



❖ Connected / Acyclic / Single-head

Ⅱ Dependency Grammar and Dependency Structure

❖ A sentence is parsed by choosing for each word what other word(including ROOT) is it a dependent of

❖ Usually some constraints:

- Only one word is a dependent of ROOT
- Don't want cycles A → B, B → A

❖ This makes the dependencies a tree

❖ non-projective / projective

Dependency Parsing

Data-driven DP

- ❖ Transition-based
- ❖ Graph-based

- assume that any input string is a valid sentence
- return the most plausible dependency structure for the input

Grammar-based DP

- ❖ Context-free
- ❖ Constraint-based

- make use of a formal grammar that only accepts a subset of all possible input strings

Data-driven Dependency Parsing

❖ Supervised dependency parsing

- Learning
  - Given a training set D of sentences (annotated with dependency graphs), induce a parsing model M that can be used to parse new sentences.

- Parsing
  - Given a parsing model M and a sentence S, derive the optimal dependency graph G for S according to M

⟹ the algorithms used to learn the model from data
and parse a new sentence with the model

| S. Kubler, et al. Dependency parsing, Morgan & Claypool, 2009.

Ⅲ Arc-standard transition-based parser

❖Three operations that we can perform (To do parsing under this transition-based)

: shift, left-arc, right-arc

- shift: buffer -> stack
- left-arc
  - ate → I: 'I' is dependent on ate
- right-arc
  - the thing that's on the top of the stack should be made a dependent of the thing that's second to top on the stack
- remove a dependent-> add new dependency(fish is a dependent of ate)

❖ parsing ends: only there's [root] in stack and buffer is empty

Ⅲ Transition-based parsing

❖ Basic transition-based dependency parser

Shift

Reduce
: create dependencies

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

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

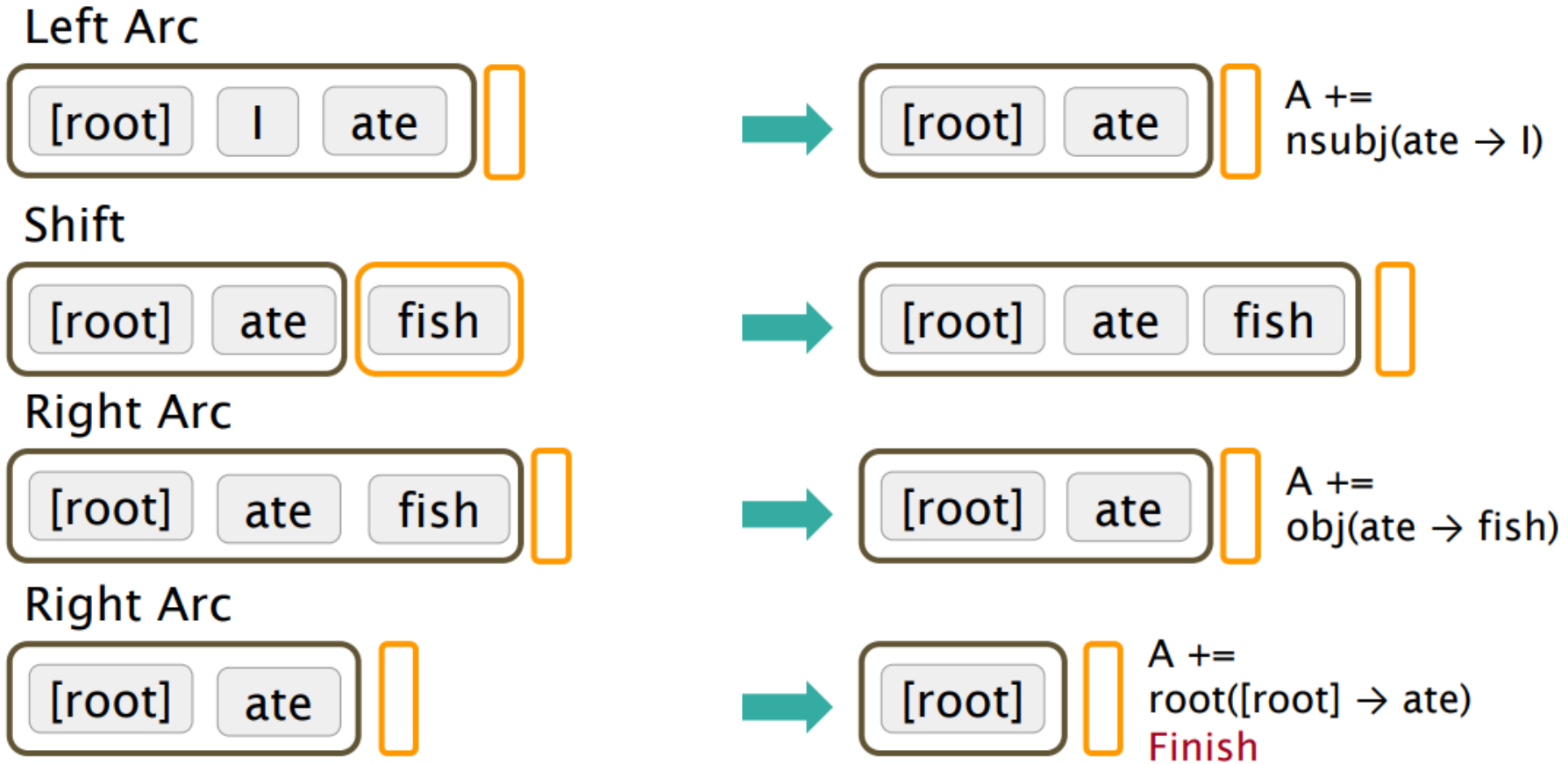
Finish: $\sigma = [w], \beta = \varnothing$

| https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1174/lecture_notes/cs224n-2017-notes4.pdf

Ⅲ Transition-based parsing

❖ Arc-standard transition-based parser

sentence : I ate fish



| https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1174/lectures/cs224n-2017-lecture6.pdf

III Neural Dependency Parsing

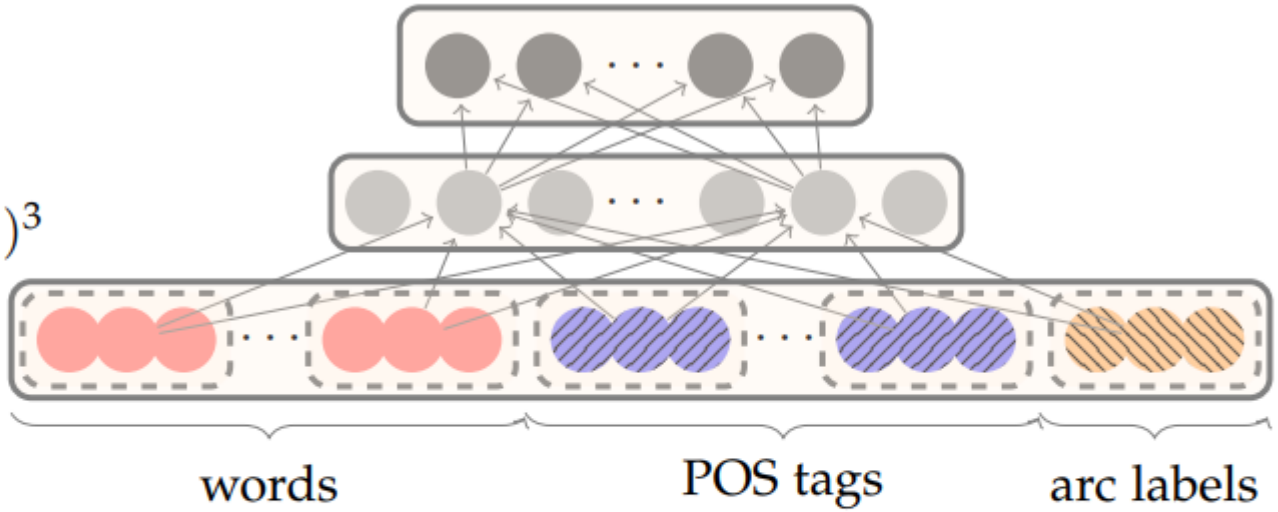
Softmax layer:

$$p = \texttt{softmax}(W_2 h)$$

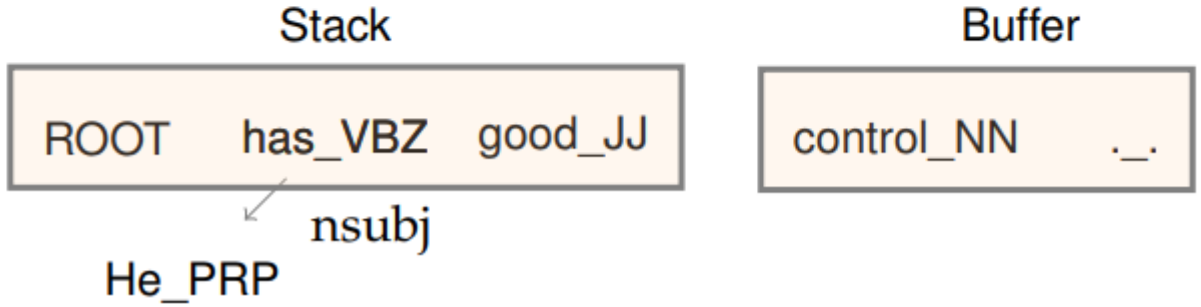
Hidden layer:

$$h = (W_1^w x^w + W_1^t x^t + W_1^l x^l + b_1)^3$$

Input layer: $[x^w, x^t, x^l]$



Configuration



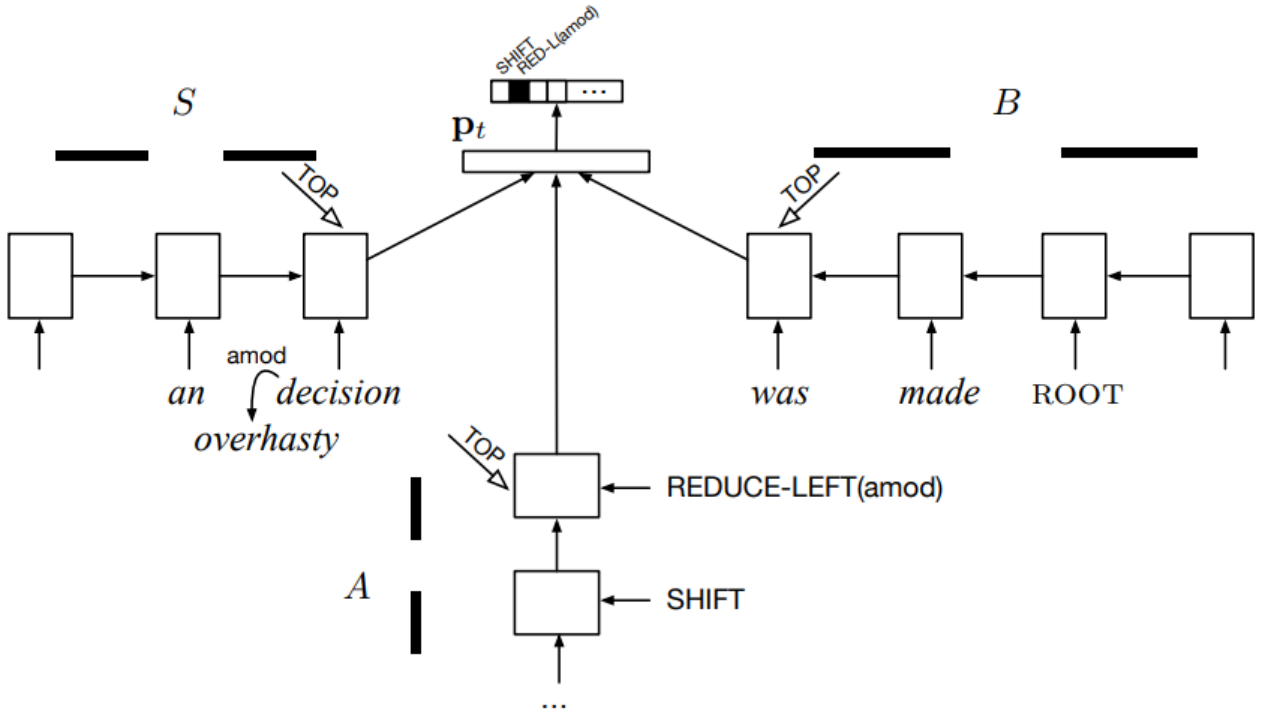
| https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1174/lecture_notes/cs224n-2017-notes4.pdf

Ⅲ Neural Dependency Parsing

❖ Transition-Based Dependency Parsing with Stack Long Short-Term Memory

$$\mathbf{p}_t = \max\{\mathbf{0}, \mathbf{W}[\mathbf{s}_t; \mathbf{b}_t; \mathbf{a}_t] + \mathbf{d}\}$$

$$p(z_t \mid \mathbf{p}_t) = \frac{\exp\left(\mathbf{g}_{z_t}^\top \mathbf{p}_t + q_{z_t}\right)}{\sum_{z' \in \mathcal{A}(S,B)} \exp\left(\mathbf{g}_{z'}^\top \mathbf{p}_t + q_{z'}\right)}$$



❖ stack에 push되는 input representation
: Recursive neural network 사용

$$\mathbf{c} = \tanh\left(\mathbf{U}[\mathbf{h}; \mathbf{d}; \mathbf{r}] + \mathbf{e}\right)$$

| C. Dyer et al. “Transition-Based Dependency Parsing with Stack Long Short-Term Memory”, ACL 2015

Ⅲ Neural Dependency Parsing

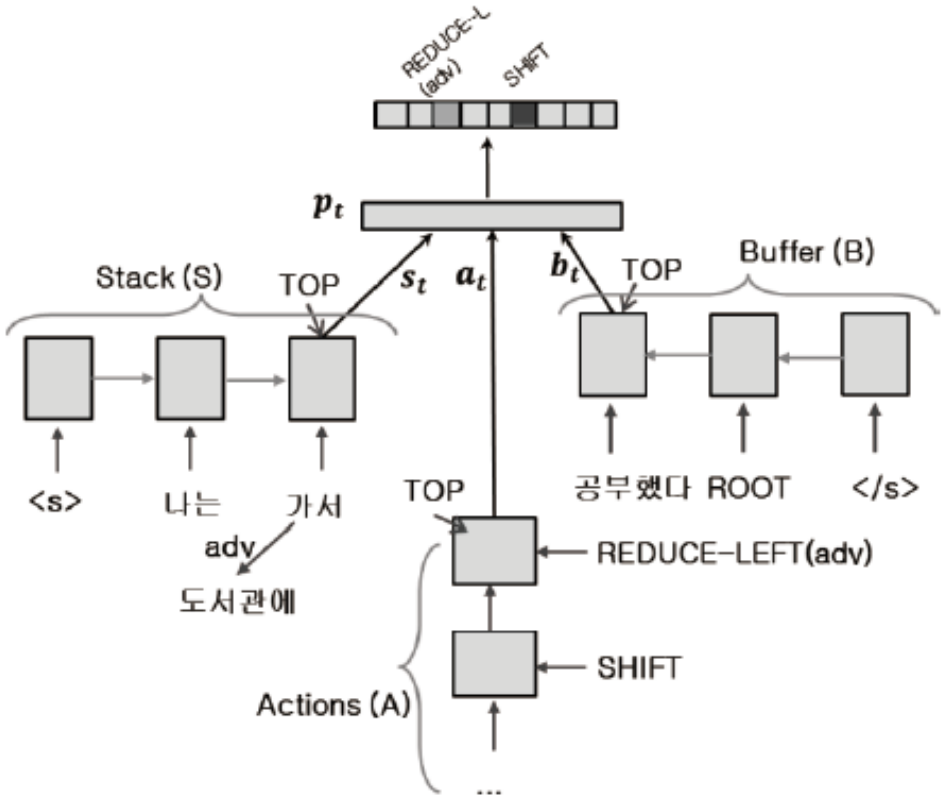
❖ 나승훈, 김상일, 김영길, “Stack LSTM을 이용한 전이 기반 한국어 의존 파싱”, KCC논문집, 2016

$$\mathbf{p}_t = \max\{\mathbf{0}, \mathbf{W}[\mathbf{s}_t; \mathbf{b}_t; \mathbf{a}_t] + \mathbf{d}\}$$

$$p(z_t \mid \mathbf{p}_t) = \frac{\exp\left(\mathbf{g}_{z_t}^{\top}\mathbf{p}_t + q_{z_t}\right)}{\sum_{z' \in \mathcal{A}(S,B)} \exp\left(\mathbf{g}_{z'}^{\top}\mathbf{p}_t + q_{z'}\right)}$$

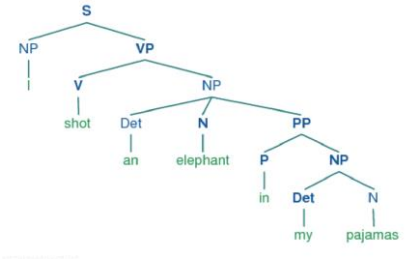
❖ stack에 push되는 input representation
: Recursive neural network 사용

$$\mathbf{c} = \tanh\left(\mathbf{U}[\mathbf{h}; \mathbf{d}; \mathbf{r}] + \mathbf{e}\right)$$

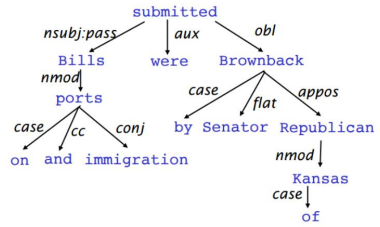


Syntactic Parsing

Constituency parsing



Dependency parsing



Data-driven DP

Grammar-based DP

Transition-based

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

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: $\sigma = [w], \beta = \varnothing$

Graph-based

Lab Seminar Plan :: Dependency parsing

❖ Week 1 (29th, Mar)

- Dependency Grammar and Dependency Structure

❖ Week 2 (5th, April)

- Dependency parsing with Deep learning(FFN) – paper research (관련 논문 정리 및 구현, 비교)
  - D. Chen and C. D. Manning, A Fast and Accurate Dependency Parser using Neural Network, EMNLP, 2014.
  - D. Andor, et al. Globally normalized transition-based neural networks. ACL, 2016.
  - 박천음, 이창기, 포인터 네트워크를 이용한 한국어 의존 구문 분석, Journal of KIISE, 2017
  - 이창기, 김준석, 김정희, 딥 러닝을 이용한 한국어 의존 구문 분석, HCLT, 2018 ~ arc-eager
- Dependency parsing with RNN
  - Sequence-to-Sequence Neural Networks based Dependency Parsing
  - Stack LSTM

❖ Week 3 (12th, April)

- Pointer networks, Stack-pointer networks
- Recent papers in regard to Korean Dependency Parsing
  - 홍승연, 나승훈, 신종훈, 김영길, "Bidirectional Stack Pointer Network를 이용한 한국어 의존 파싱", 제 30회 한글 및 한국어 정보처리 학술대회, 2018.10
  - 홍승연, 나승훈, 신종훈, 김영길, "동적 오라클을 적용한 Stack Pointer Network 기반 한국어 의존 파싱", 한국 정보과학회 동계 학술발표논문집, 2018.12
  - 홍승연, 나승훈, 신종훈, 김영길, "Multi-level Biaffine Attention을 이용한 한국어 의존 파싱", 한국 정보과학회 동계 학술발표논문집, 2018.1

감 사 합 니 다.

남궁영
한국해양대학교 컴퓨터공학과
young_ng@kmou.ac.kr