Lec 4. Syntactic Structure and Dependency Parsing

1. Syntactic Structure: Consistency and Dependency

-Two views of linguistic structure:

*Constituency=Phrase

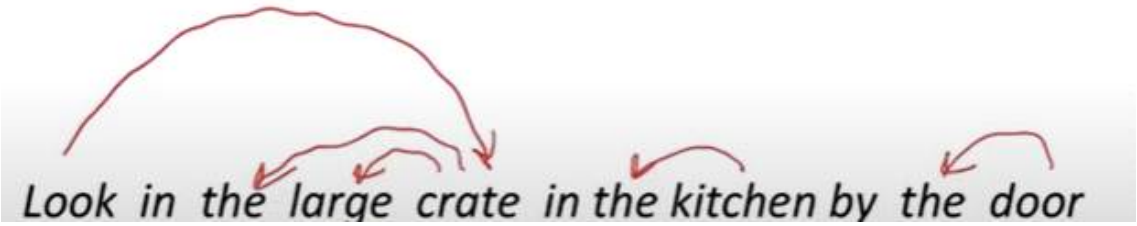
structure grammar = context-free grammars(CFGs)

eg) VP -> V, pp

S -> NP VP

*Dependency structure

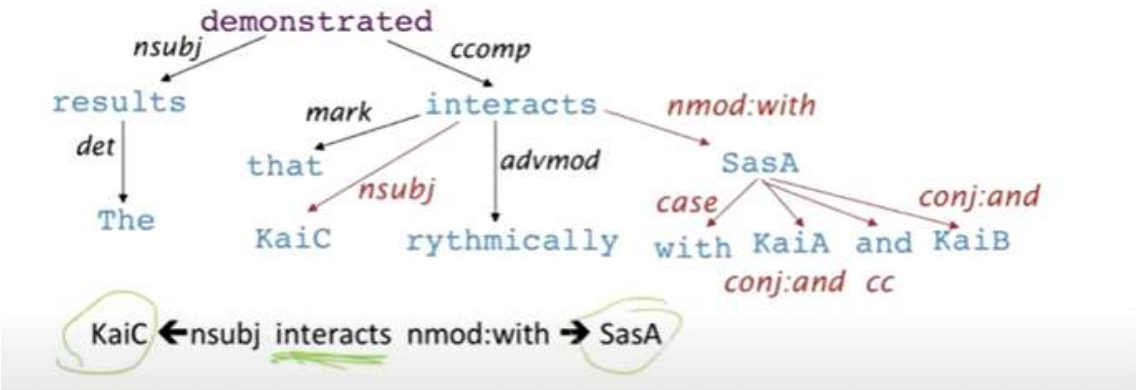
shows which words depend on (modify, attach to, or are arguments of) which other words -> neural dependency parser



단어의 중의성, 모호성 등의 문제들 ...

뉴스제목은 특히 더 모호한 경향있음

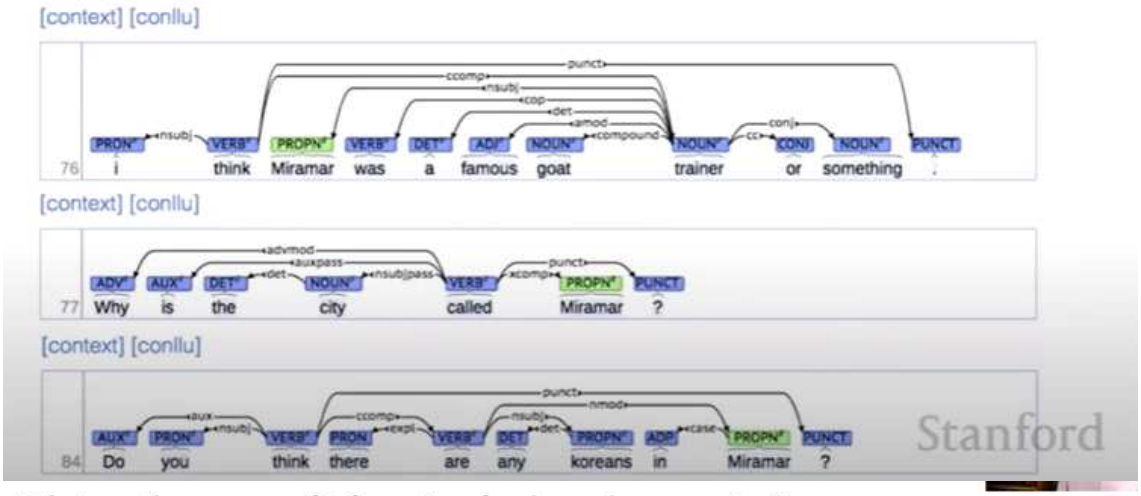
Modifier Ambiguity/VP attachment ambiguity



Dependency paths help extract semantic interpretation

*fake root

2. Dependency Grammar and Treebank



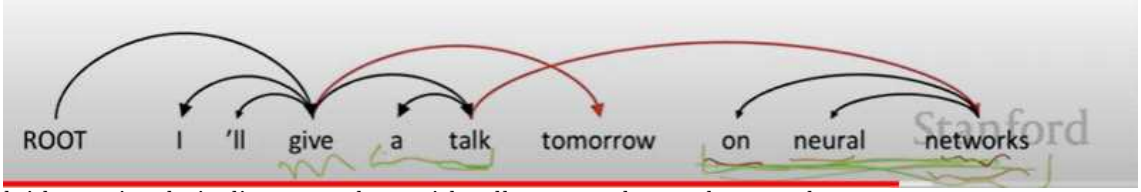
What are the sources of information for dependency parsing?

1. Bilexical affinities — The dependency [discussion → issues] is plausible
2. Dependency distance — Most dependencies are between 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?

The rise of annotated data & Universal Dependencies tree
slower and less useful than writing a grammar
-> but, Reusabillity of the labor, broad coverage, not just a few intuitions, frequencies and distrivutional infrom, a way to evaluate NLP systems

3. Transition-based dependency parsing
-Dependency Parsing
A sentence is parsed by choosing for each word what other word(including ROOT) it is 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
Final issue is whether arrows can cross (be non-projective) or not

-Projectivity
<projective parse>: There are no crossing dependency arcs when the words are



laid out in their linear order, with alla arcs above the words.
Dependencies corresponding to a CFG tree must be projective.

1. Dynamic programming
2. Graph algorithms
3. Constraint Satisfaction
4. “Transition-based parsing” or “deterministic dependency parsing”

-Greedy transition-based parsing: a simple form of greedy discriminative dependency parser. bottom-up actions

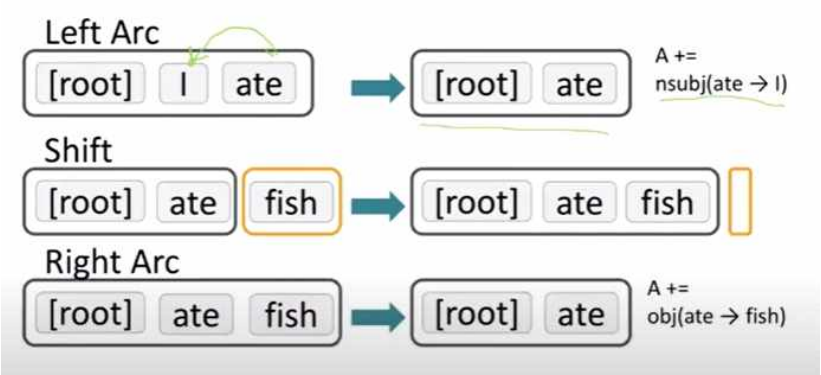
- The parser has:
  - a stack σ, written with top to the right
    - which starts with the ROOT symbol
  - a buffer β, written with top to the left
    - which starts with the input sentence
  - a set of dependency arcs A
    - which starts off empty
  - a set of actions

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

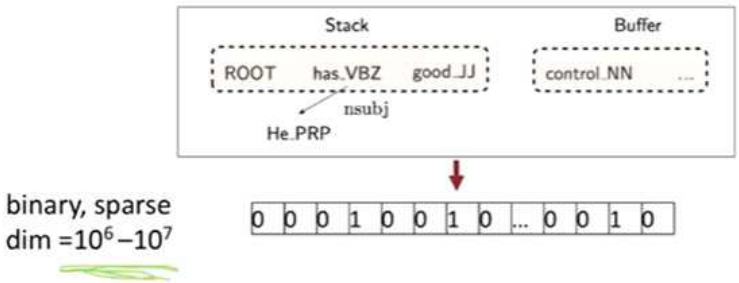
1. Shift $\sigma, w_i|\beta, A \rightarrow \sigma|w_i, \beta, A$
2. Left-Arc$_r$ $\sigma|w_i|w_j, \beta, A \rightarrow \sigma|w_j, \beta, A \cup \{r(w_j,w_i)\}$
3. Right-Arc$_r$ $\sigma|w_i|w_j, \beta, A \rightarrow \sigma|w_i, \beta, A \cup \{r(w_i,w_j)\}$

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

-MaltParser -> 다음 action 어떻게 정할 것인지?
-> Stand back, macine learning, each action os predicted by a discriminative Classifier(eg. softmax classifier), no search(in the simple form), model’s accuacy -> fractionally below the state of the art in dependency parsing but very fast linear time parsing with high accuracy- good for web!



Conventional Feature Representation



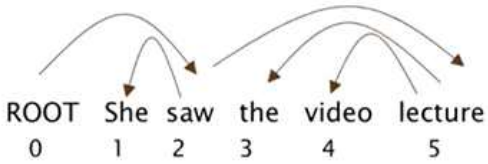
Indicator features

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

Stanford

dependency accuracy

Evaluation of Dependency Parsing: (labeled) dependency accuracy



Acc = $\frac{\text{\# correct deps}}{\text{\# of deps}}$

UAS = 4 / 5 = 80%

Gold			
1	2	She	nsubj
2	0	saw	root
3	5	the	det
4	5	video	nn
5	2	lecture	obj

Parsed			
1	2	She	nsubj
2	0	saw	root
3	4	the	det
4	5	video	nsubj
5	2	lecture	ccomp