Lecture 4 - Syntactic Structure and Dependency Parsing

1. Syntactic Structure : Consistency and Dependency

- **Consistency**
  - constituency = phrase
  - structure grammar = context-free grammars
  - **Phrase structure** organizes words into nested constituents

- **Dependency**
  - Dependency structure shows which words depend on (modify, attach to, or are arguments of) which other words.

- Ambiguity
  1. Phrase attachment ambiguity
  2. Coodination Scope ambiguity

2. Dependency Grammar and Treebanks

- Dependency syntax postulates that syntactic structure consists of relations between lexical items, normally binary asymmeric relations “arrows” called dependencies
  - An arrow connects a head (governor, superior, regent) with a dependent (modifier, inferior, subordinate)
  - Usually, dependencies form a tree (a connected, acyclic, single-root graph)

- Dependency Parsing

    - A sentence is parsed by choosing for each word what other word it is a dependent of

- Projectivity
    - There are no crossing dependency arcs when the words are laid out in their linear order, with all arcs above the words

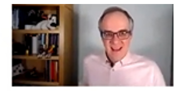
- The rise of annotated data : treebank’s advantages
    - Reusability of the labor
    - Broad coverage
    - Frequencies and distributional information
    - A way to evaluate NLP system

- Dependency Conditioning Preferences
    1. Bilexical affiniteis
    2. Dependency distance
    3. Intervening material
    4. Valency of heads

3. Transition-based dependency parsing

- Methods of Dependency Parsing
    1. Dynamic programming
    2. Graph algorithms
    3. Constraint Stisfaction
    4. “Transition-based parsing” or “deterministic dependency parsing”
        - Greedy transition-based parsing [Nivre 2003]

Greedy transition-based parsing [Nivre 2003]



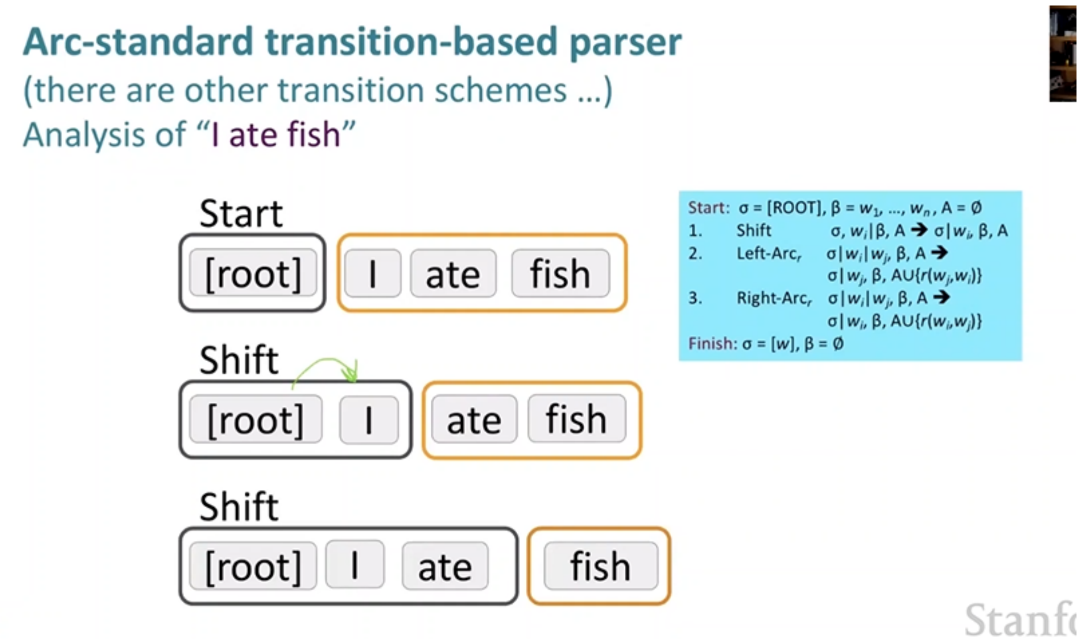
- A simple form of greedy discriminative dependency parser
- The parser does a sequence of bottom-up actions
  - Roughly like “shift” or “reduce” in a shift-reduce parser, but the “reduce” actions are specialized to create dependencies with head on left or right
- The parser has:
  - a stack σ, written 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

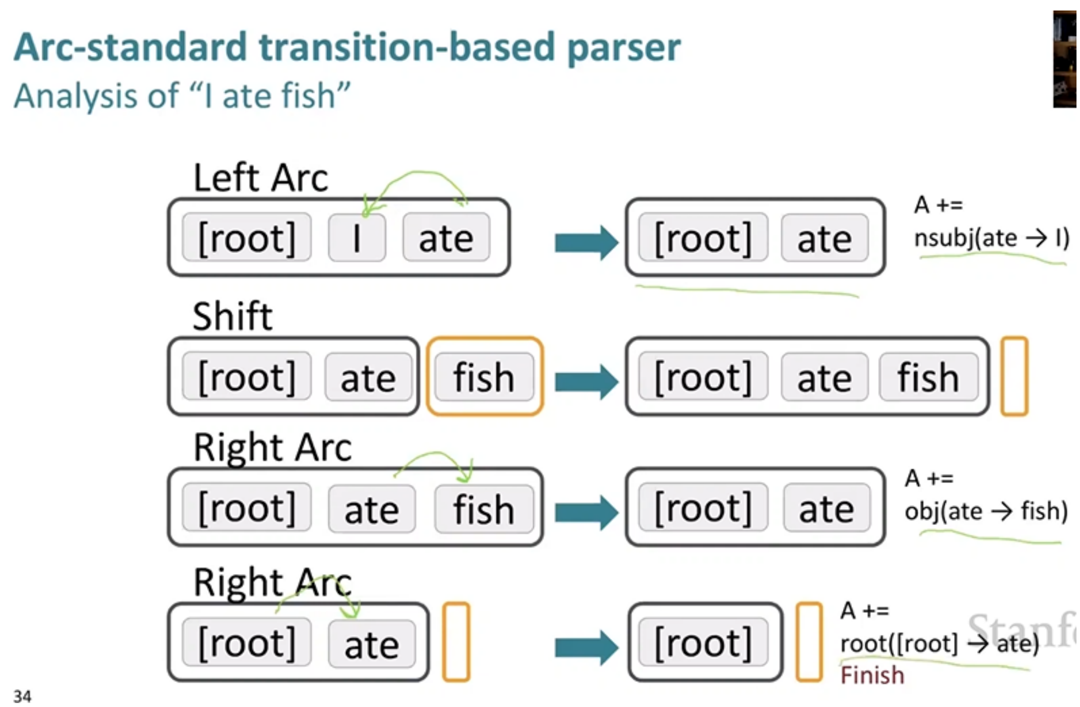
Basic transition-based dependency parser

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

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

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



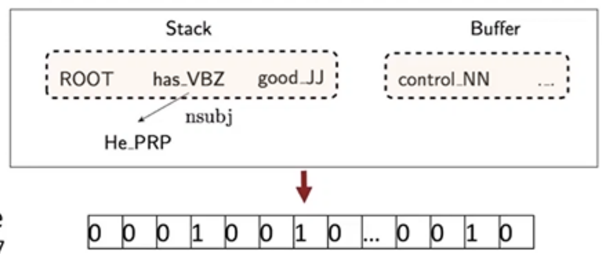


- MaltParser [Nivre and Hall 2005]

MaltParser [Nivre and Hall 2005]

- We have left to explain how we choose the next action
  - Answer: Stand back, I know machine learning!
- Each action is predicted by a discriminative classifier (e.g., softmax classifier) over each legal move
  - Max of 3 untyped choices; max of |R| × 2 + 1 when typed
  - Features: top of stack word, POS; first in buffer word, POS; etc.
- There is NO search (in the simplest form)
  - But you can profitably do a beam search if you wish (slower but better): You keep *k* good parse prefixes at each time step
- The model's accuracy is *fractionally* below the state of the art in dependency parsing, but
- It provides **very fast linear time parsing**, with high accuracy – great for parsing the web

Conventional Feature Representation



binary, sparse
dim $=10^6 – 10^7$

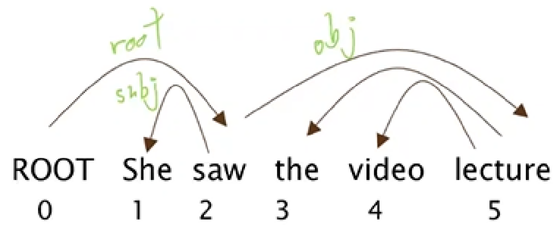
Indicator features

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

36

- Evaluation of Dependency Parsing

Evaluation of Dependency Parsing: (labeled) dependency accuracy



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

UAS = 4 / 5 = 80%

LAS = 2 / 5 = 40%

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

Stanfc