Lecture 7 Dependency Parsing

Linguistice Structure

Sytactic Amigiuities

- Grammars are declarative

  — They don't specify how the parse tree will be constructed

- Ambiguity

  — Prepostional phrase(PP) attachment ambiguity
  - a key parsing decision is how we attach various constitutents

  — Verb attachment ambiuity

  — Anaphoric ambiguity

  — Coordination Scope
  - 'I ate [red (apples] and bananas)'

  — Particles pr Prepositions
  - some verbs are followed by adverb particles, and some particles are detached from the verb and put after the noun
  - adverb particle: partic is closely tied to its verb to form idiomatic expressions
  - Preposition is closely tied to the noun or pronoun it modifies.

  — Gerund or adjective
  - ' Dancing shoes can provide noce experience'

  — Gap
  - 'She nerver saw a dog and did not smile '

- Views of linguistic structure

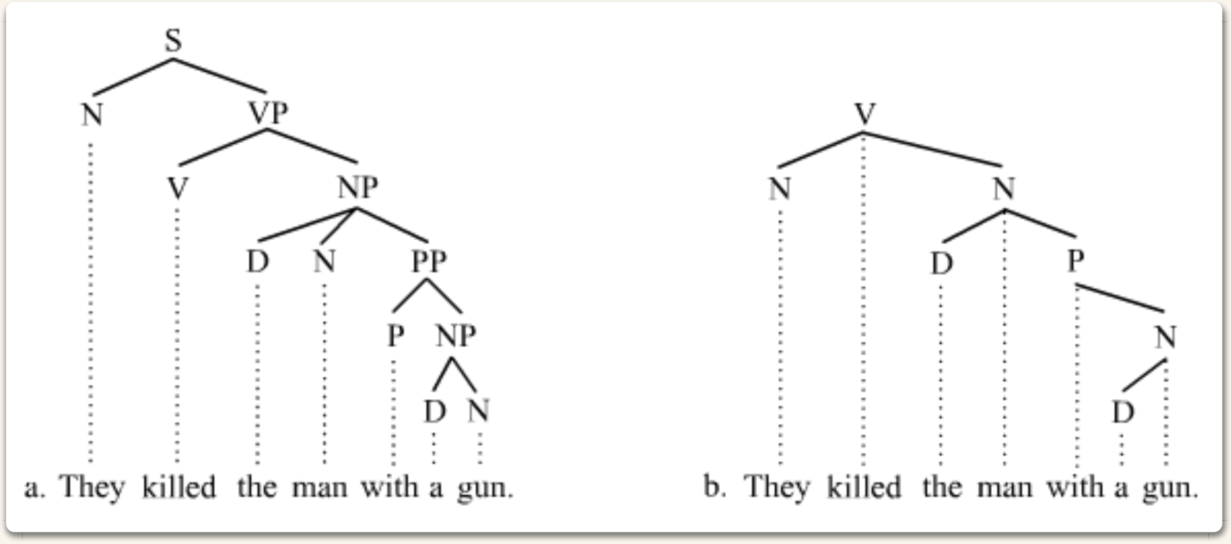
— Constituency Grammar

- Immediate constituent analysis
- one to one/more relation.
- For every word in a sentence, there is at least one node in the suntactic structure that corresponds to that word
- a basic observatione: groups of word (constituents) = a single unit
- constituents: similar internal structure and behave similarity with respect to other units
- e.g. noun phrases(NP) 、 verb phrases(VP)、prepppositional phrases(PP)

— Dependencu Grammar

- Functional dependecy relations
- For every word in a sentence, there is exactly one node in the syntactic structure that corresponds to that word.

Constituency Parsing Dependency Parsing



- Sample context-free grammar

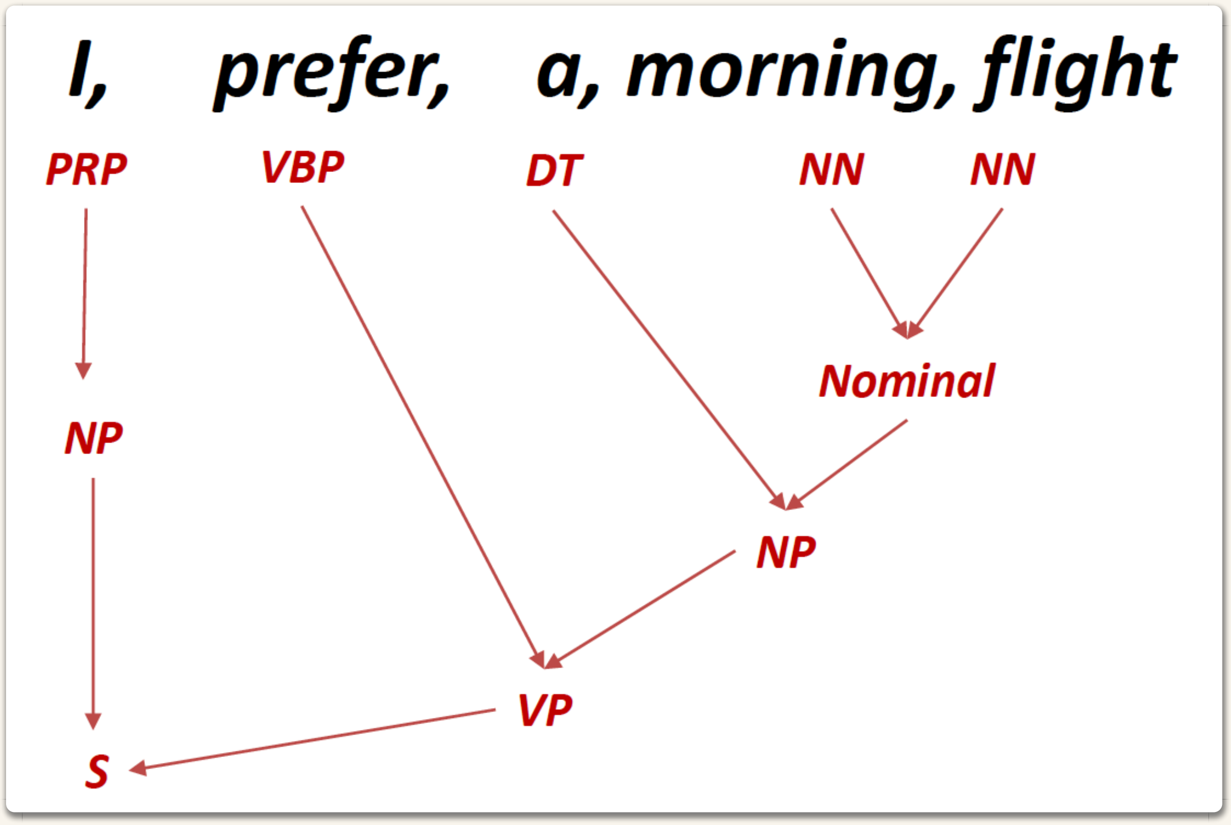
— Starting unit: words are give a category (PoS)

I, prefer, a, morning, flight

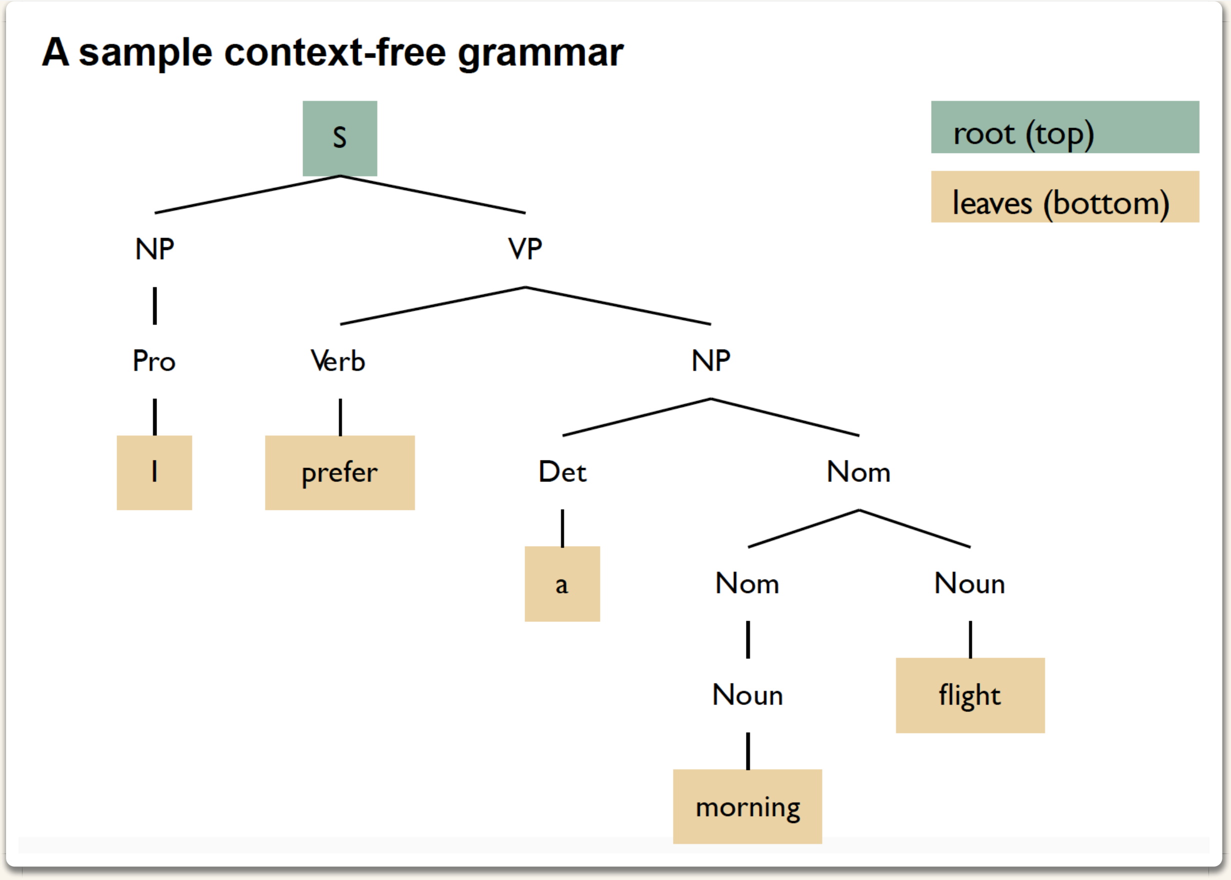
PRP, VBP, DT, NN, NN

Grammar rule	Example
S → NPVP	I + want a morning flight
NP → Pronoun	I
NP → Proper-Noun	Sydney
NP → Det Nominal	a flight
Nominal → Nominal Noun	morning flight
Nominal → Noun	flights
VP → Verb	do
VP → Verb NP	want + a flight
VP → Verb NP PP	leave + Melbourne + in the morning
VP → Verb PP	leaving + on Thursday
PP → Preposition NP	from + Sydney

— Combining words into phrases with categories



— combining phrases into bigger phrases recursively



- Treebanks
  - Treebanks are generally created by

    — parsing texts with an existing parser

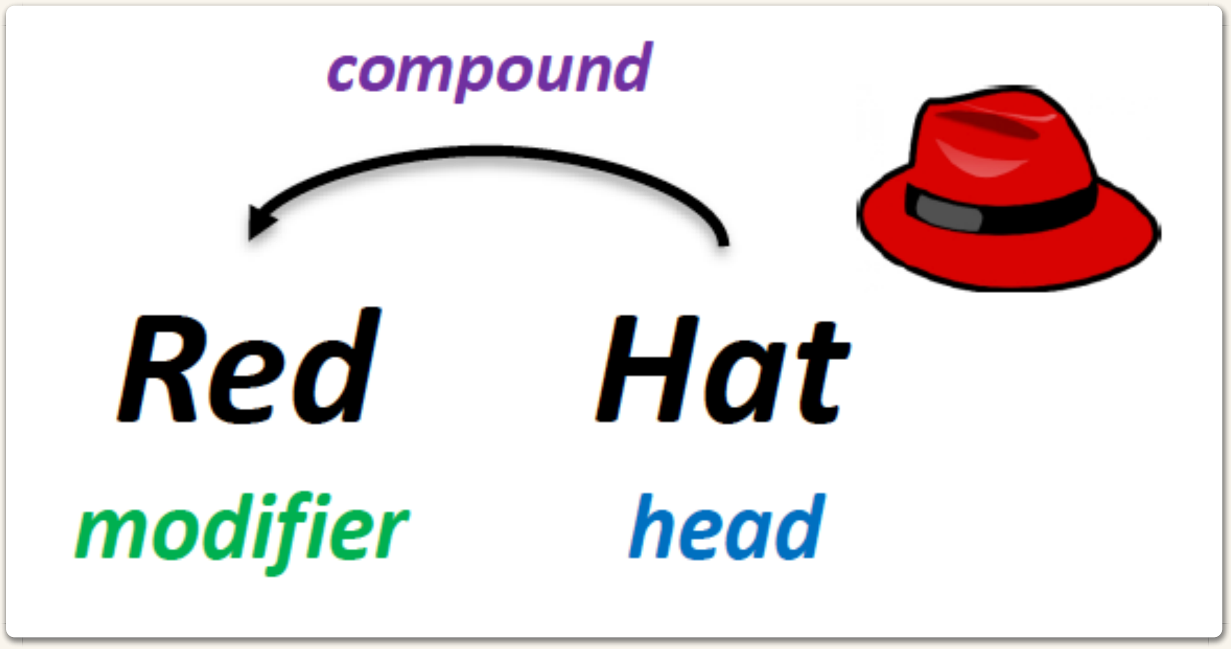
    — having human annotators correct the result

  - Penn Treebank (English annotators)

```
( (S
    (NP-SBJ
      (NP (NNP Pierre) (NNP Vinken) )
      (, ,)
      (ADJP
        (NP (CD 61) (NNS years) )
        (JJ old) )
      (, ,) )
    (VP (MD will)
      (VP (VB join)
        (NP (DT the) (NN board) )
        (PP-CLR (IN as)
          (NP (DT a) (JJ nonexecutive) (NN director) ))
        (NP-TMP (NNP Nov.) (CD 29) )))
    (. .) ))
```

Dependency Structure

- Syntactic structure : Dependncies, lexical items linked by binary asymmetrical relations("arrows")



— modeifier: dependent, child,subordinate

— head: governor, parent,regent

determine the syntactic/semantic category of the construct

—compound: dependency relations

The arrows are typed with the name of grammatical relations

- Dependency Parsing

— Represents lexical/syntactic dependencies between words

— Dependencies from a tree (connected, acyclic,single-head)

  - How to make the dependencies a tree- Constraints
    1. only one wordis a dependent of ROOT(the main predicate of a sentence)
    2. Don't want cycles A—>B,B—>A

3. Usally add one fake ROOT to avoid the difference among ways of the drawing arrows, so every word is a dependent of precisely one other node
4. Projevtivity vs Non-projectivity

   — No crossing dependency arcs when the words are laid out in their linear order, with all arcs above the words

   — Dependencies parallel to a Context-Free-Grammar(CFG) tree must be projective
   - Forming dependencies by taking 1 child of each category as head

   — Dependency theory does allow non-projective structures to account for displaces constituents.

— Advantage: Treebank
- Reusability of the labor: many parsers,PoS taggers, valuable resource for linguistics
- Broad coverage, not just a few intuitions
- Frequency and distributional information
- A way to evaluate systems

- Dependency Conditioning Preference

— Source of information for dependency parsing
- Bilexical affinities
- Dependency distanec
- Intervening material
- Valency of heads

Dependency Parsing Algorithms

- Dynamic programming

— $O(n^3)$: producing parse items with heads at the ends rather than in the middle
- Constraint Satisfication

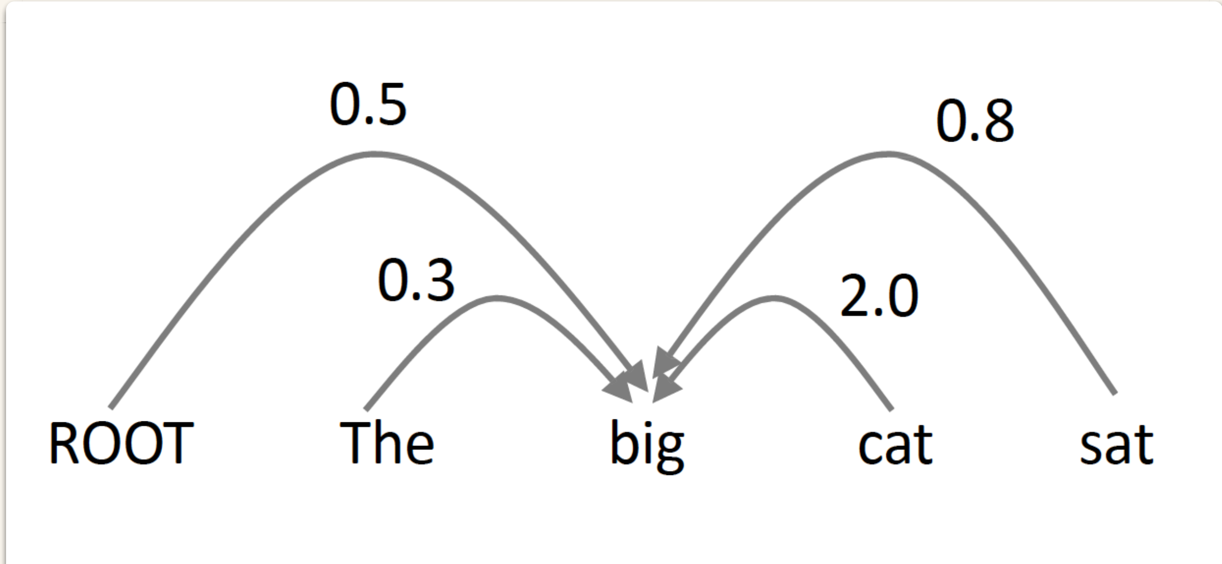
— edges are eliminated that don't satisfy hard constraints

- Graph-based Dependency parsing (MST)

  — Non-deterministic dependency parsing
  - build a complete graph with directed weighted edges
  - Find the highest scoring tree from a couplete dependency graph

  — create a Minimum Spanning Tree for a sentence
  - computing a score for every possible dependency for each edge
  - Add an edge from each word to the highest scoring candidate head
  - Repeat the same process for each other word

  — MST parser scores dependencies independently using a ML classifier



e.g., picking the head for "big"

- Transition-based Dependency parsing (Greedy Algorithm)

  — Deterministic dependecy parsing
  - build a tree by applying a sequence of transition actions
  - Find the highest scoring action sewuence that builds a legal tree

- Neural Network-based Dependency Parsing

Transition-based Parsing

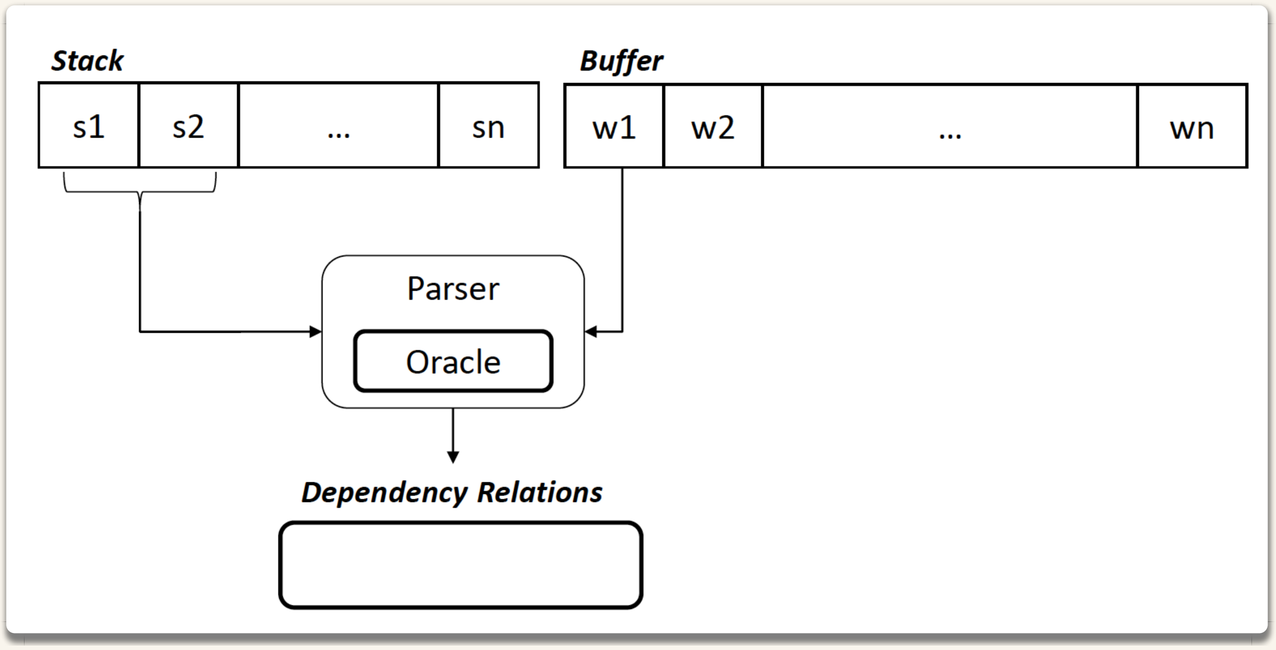
Greeding Transition-based parsing

— A form of greedy discriminatiove dependecy parser

— Design a dumber but really fast algorithm and let the machine learning do the rest

— Eisner's Algorithm (Dynamic Programming–based Dependency parsing) searches over many different dependency trees at the same time

— a transition–bsed dependency parser only builds one tree, in one left–to–right sweep over the input



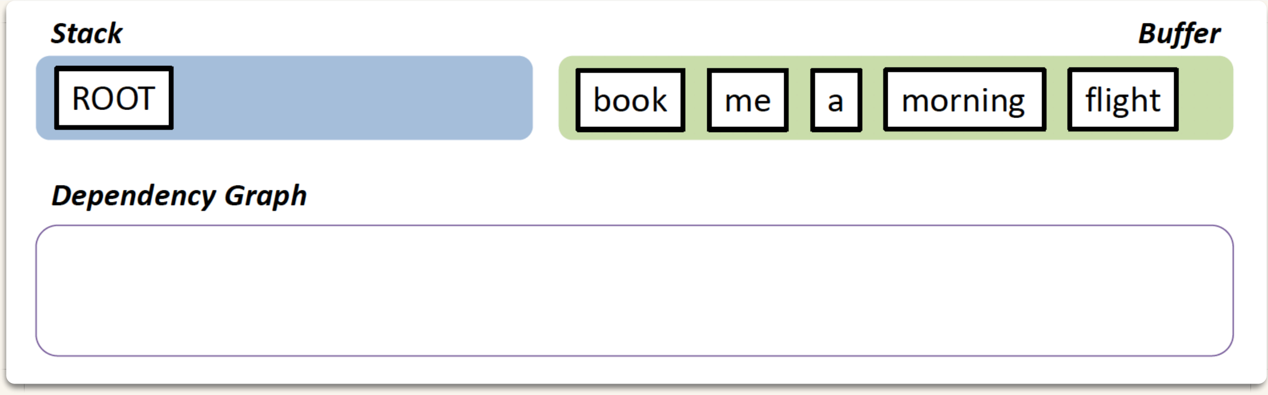
Transition–based parsing– the arc–standard algorithm

— The arc–standard algorithm is a simple algorithm for transition–based dependency parsing

— A sequence of bottom up actions:

- Like "shift" or "reduce" in a shift–reduce parser,but the "reduce" actions are specialized to create dependencies with head on left or right

— Most prctical transition–based dependency parser including MaltParser



- Initial configuration
  - All words are in the buffer
  - the stack is empty or starts with the ROOT symbol
  - the dependency graph is empty
  - ROOT cannot have incoming arc
- Possible Transaction
  - Shift: Push the next word in buffer onto the stack
  - Left–Arc:
    - Add an arc form the topmost word to the 2^{nd} topmost word on the stack
    - Remove 2^{nd} word from stack
    - Require 2 elements in stack to be applied
  - Right–Arc:
    - Add an arc from the 2^{nd}–topmost word to the topmoset word on the stack
    - Remove the topmost word from stack
    - Require 2 elements in stack to be applied
- Terminal configuration
  - The buffer is empty
  - The stack contains a single word
-

Start: σ = [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$_r$ $\sigma|w_i|w_j, \beta, A \rightarrow \sigma|w_j, \beta, A \cup \{r(w_j,w_i)\}$ $\sigma|w_i|w_j, \beta, A \rightarrow$
3. Right-Arc$_r$ $\sigma|w_i, \beta, A \cup \{r(w_i,w_j)\}$

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

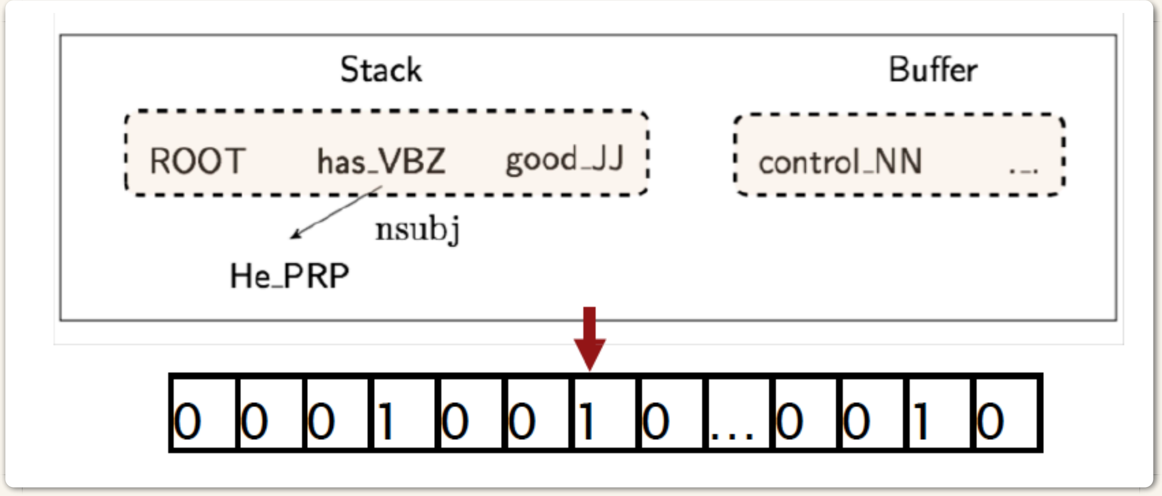
-

- choose the next action (ML)
  - Goal: predict the next transition(class), given the current configuration
  - Let the parser run on gold-standard trees
  - Every time there is a choice to make, we just looke into the tree to do the right thing
  - Collect all (configuration,transition) pairs and train a classifier on them
  - When parsing unseen senteces, we use the trained class classifier as a guide

- Handling the large number of pairs: Feature representation
  - define a set of features of configurations that would be relevant for the task of predicting athe next transition.

    — e.g. word forms of the topmost two words on the stack and the next two words in the buffer

  - Describe every configuration in terms of a feature vector



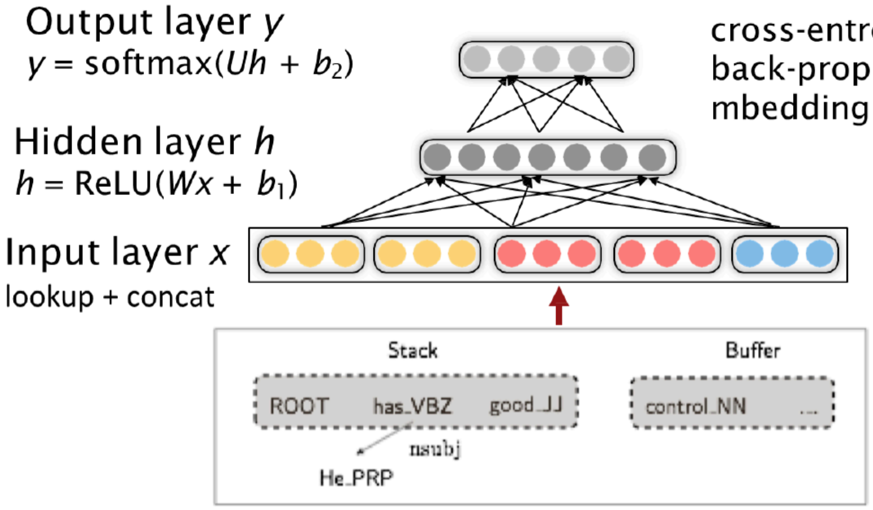
- In practics, thousands of features and hundrends of transitions
- use ML (preceptor,decision tree, Sam, memory-based learning)

Deep Learning-based Dependency parsing

Distributed Representations

- Represent each word as a d-dimensional dense vector (word embedding)
  - Similar words are expected to have close vector
  - NNS (plural noun) should be close to NN(singular noun)
- PoS and dependency labels are also represented as the d-dimensional vector
- Similar discrete sets also represents similar semantic
-

Softmax probabilities



cross-entropy error will be back-propagated to the e mbeddings.