

Natural Language Processing

Dependency Parsing

Sudeshna Sarkar

18 Mar 2020

What is Syntax?

- Structure of language
 - How words are arranged together and related to one another
 - Ordering words in sequences to express meanings for which no separate word exists.
- Goal of syntactic analysis
 - relate surface form to underlying structure, to support semantic analysis
- Two views of linguistic structure:
 1. Constituency = phrase structure grammar= context-free grammars (CFGs)
 2. Dependency Structure
 - Dependency structure shows which words depend on (modify or are arguments of) which other words.

Why is Syntax Important?

- Grammar checkers
 - Question answering
 - Information extraction
 - Machine translation
 - Semantic role labeling
- Many aspects of meaning can be learnt using the syntactic structure.
 - The NP preceding VP is likely the subject of the action.
 - The NP following the VP is likely the object of the action.
 - Knowing basic units is helpful in modeling language.
 - You can use this to predict or complete the sentence.
 - Re-organize sentences or simplify them.

Phrase Structure Grammar

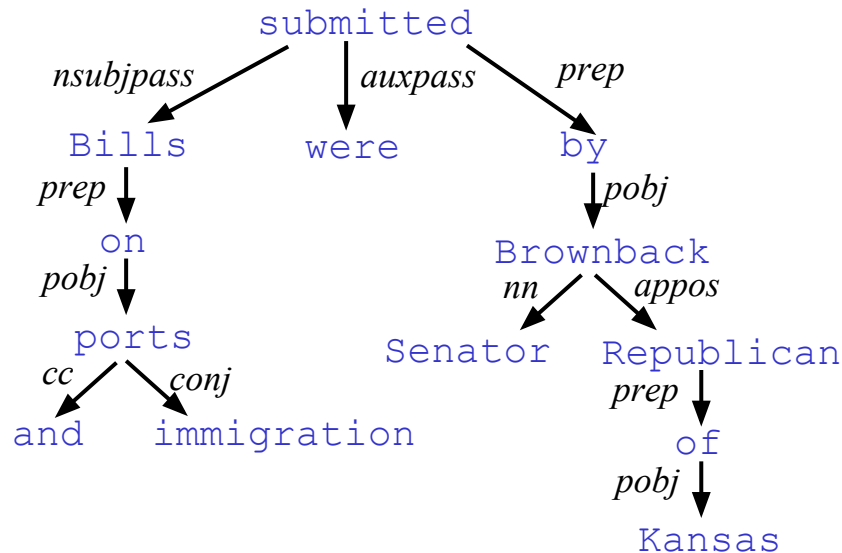
- Phrase structure organizes words into nested constituents - Can represent the grammar with CFG rules
- **Starting unit: words** are given a category (part of speech = pos)
the, cat, cuddly, by, door
Det N Adj P N
- **Words combine into phrases** with categories
the cuddly cat, by the door
 $NP \rightarrow \text{Det Adj N}$ $PP \rightarrow P NP$
- **Phrases can combine into bigger phrases** recursively
the cuddly cat by the door
 $NP \rightarrow NP PP$

Dependency Grammar and Dependency Structure

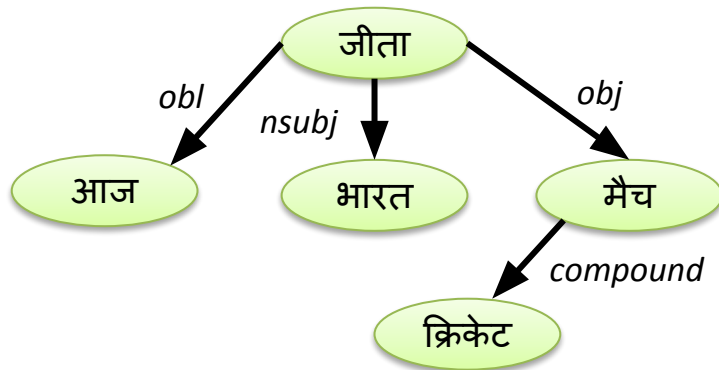
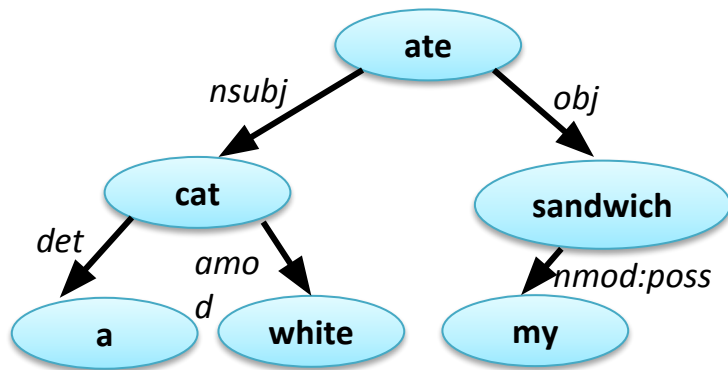
Dependency syntax postulates that syntactic structure consists of lexical items linked by binary asymmetric relations (“arrows”) called dependencies

The arrows are commonly **typed** with the name of grammatical relations

The arrow connects a head (governor) with a dependent (modifier)



Dependency Trees



Exercise: Draw dependency trees for the following sentences

1. *Look for the large barking dog by the door in a crate*
2. *Scientists study whales from space*

Dependency Structure

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

1. *Look for the large barking dog by the door in a crate*
2. *Scientists study whales from space*

Pāṇini's grammar (c. 5th century BCE)



Dependency Relations

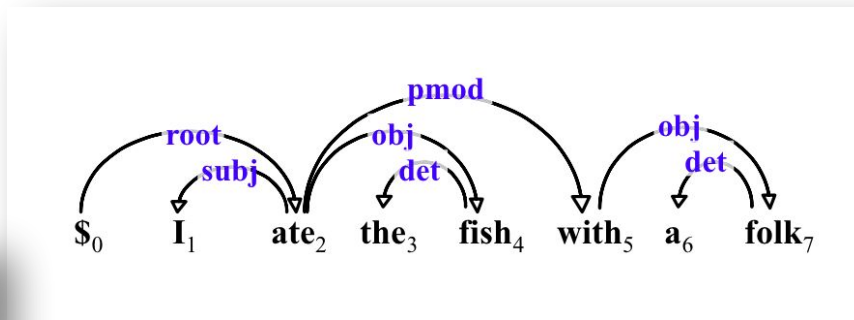
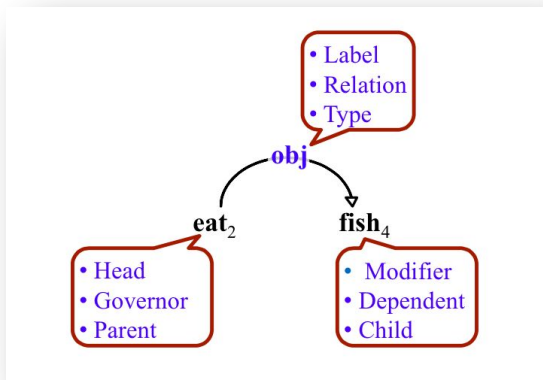
(A sample from UDEP)

Clausal Argument Relations	Description
NSUBJ	Nominal subject
DOBJ	Direct object
IOBJ	Indirect object
CCOMP	Clausal complement
XCOMP	Open clausal complement
Nominal Modifier Relations	Description
NMOD	Nominal modifier
AMOD	Adjectival modifier
NUMMOD	Numeric modifier
APPOS	Appositional modifier
DET	Determiner
CASE	Prepositions, postpositions and other case markers
Other Notable Relations	Description
CONJ	Conjunct
CC	Coordinating conjunction

Dependency Parsing

- **Dependency tree** – A tree composed of the input words, which meets a few constraints:

- Single-head
- Connected
- Acyclic

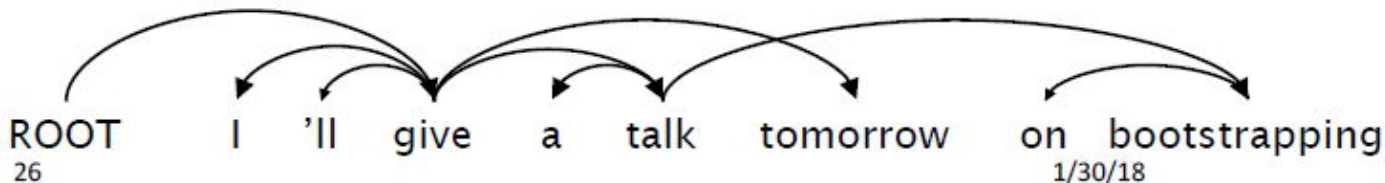


Projective Parse:

Arcs don't cross each other.
Mostly true for English.

Non-projective Parse:

Common in lang. w/ more flexible word order.
German, Dutch, Czech etc.

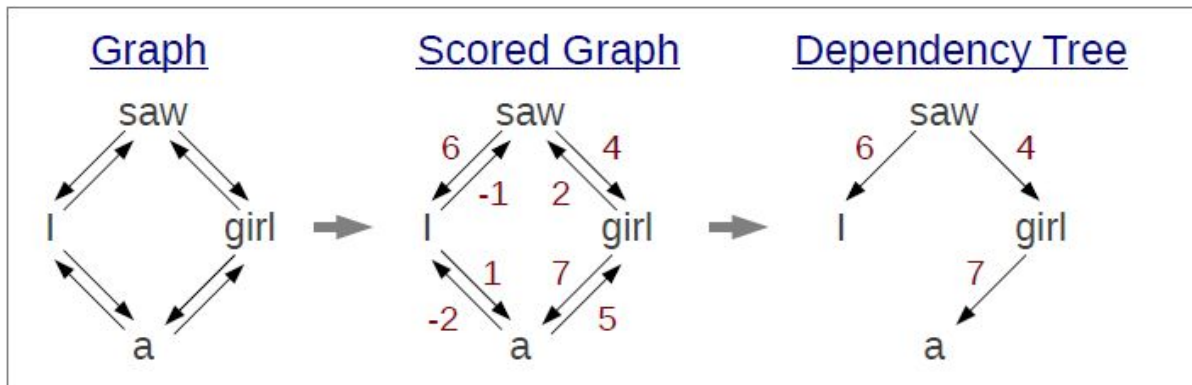


Parsing algorithms

- Dependency based parsers can be broadly categorized into
 - Grammar driven approaches: Parsing done using grammars
 1. Dynamic programming - Eisner (1996), McDonald (2006)
 2. Deterministic Search - Covington (2001), Nivre (2006)
 3. Constraint satisfaction
 - Data driven approaches: Parsing by training on annotated/un-annotated data
 1. Could construct a fully connected tree, and prune it.
 - Graph-based methods
 2. Could construct a tree one edge at a time.
 - Transition parsing.

Maximum Spanning Tree

- Each dependency is an edge in a directed graph
- Assign each edge a score (with machine learning)
- Keep the tree with the highest score



(Chu-Liu-Edmonds Algorithm)

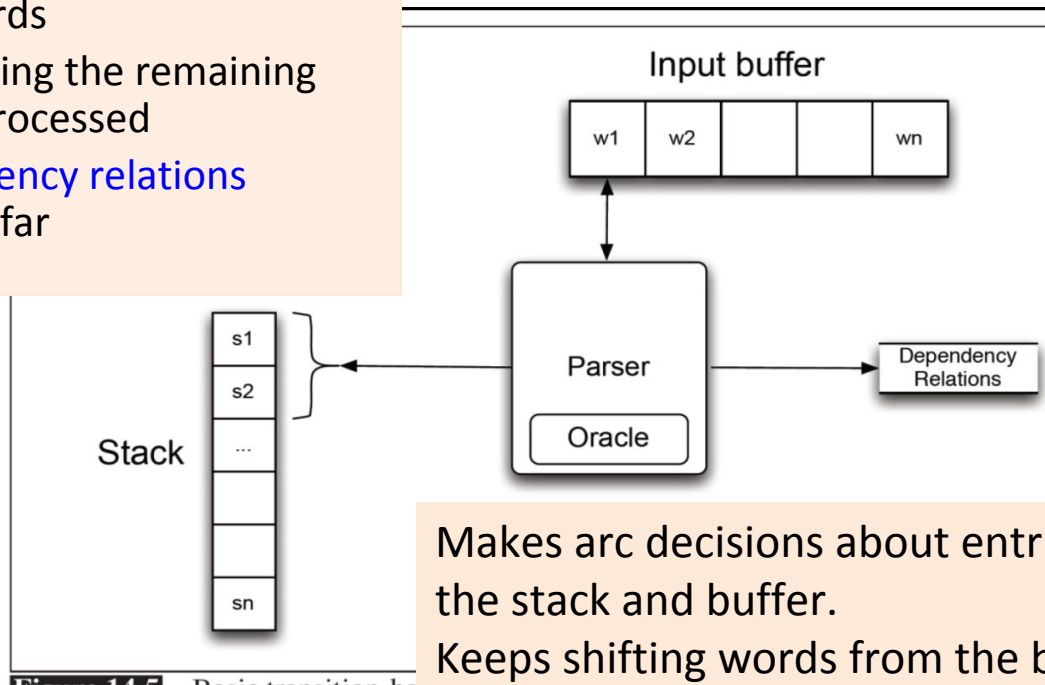
Transition-based parsing

- Transition-based parsing is a greedy word-by-word approach to parsing
 - A single dependency tree is built up an arc at a time as we move left to right through a sentence
 - No backtracking
 - ML-based classifiers are used to make decisions as we move through the sentence
- A transition system for dependency parsing defines:
 - a set C of parser configurations or states
 - a set T of transitions, each a function $t: C \rightarrow C$

Transition-based dependency parsing

Configuration or State

- **Stack** representing partially processed words
- **Buffer** containing the remaining words to be processed
- **Set of dependency relations** discovered so far



Makes arc decisions about entries in the top of the stack and buffer.

Keeps shifting words from the buffer until all words are consumed.

States

- Start state:

`[[root], [word list], ()]`

`[[root], [I booked a morning flight], ()]`

- A valid final state:

`[[root], [] (R)]`

`[[root],`

`[],`

`((booked, I) (booked, flight) (flight, a) (flight, morning)))]`

- Empty buffer
- R is the set of relations that we've discovered

Arc Standard Transition System

Defines 3 transition operators

- **LEFT-ARC:** $\sigma, w_i | w_j, \beta, A \rightarrow \sigma | w_j, \beta, A \cup \{r(w_j, w_i)\}$
 - create head-dependent rel. between word at top of stack and 2nd word (under top)
 - remove 2nd word from stack
- **RIGHT-ARC:** $\sigma, w_i | w_j, \beta, A \rightarrow \sigma | w_i, \beta, A \cup \{r(w_i, w_j)\}$
 - Create head-dependent rel. between word on 2nd word on stack and word on top
 - Remove word at top of stack
- **SHIFT** $\sigma, w_i | \beta, A \rightarrow \sigma | w_i, \beta, A$
 - Remove word at head of input buffer
 - Push it on the stack

Note: There are other transition schemes such as Arc-eager

Example: Arc-standard

Analysis of “I went home”

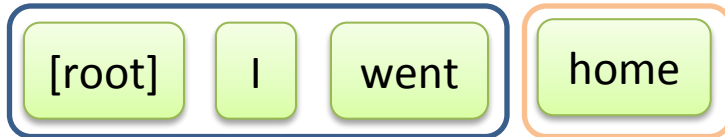
Start



Shift

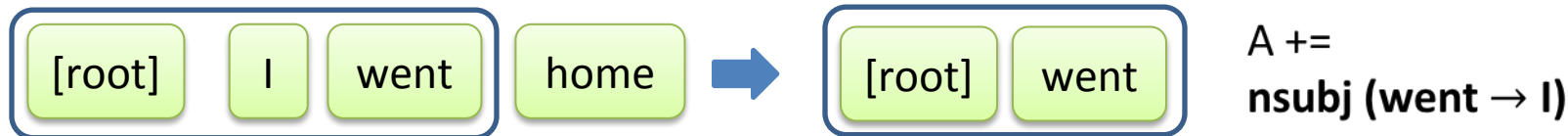


Shift

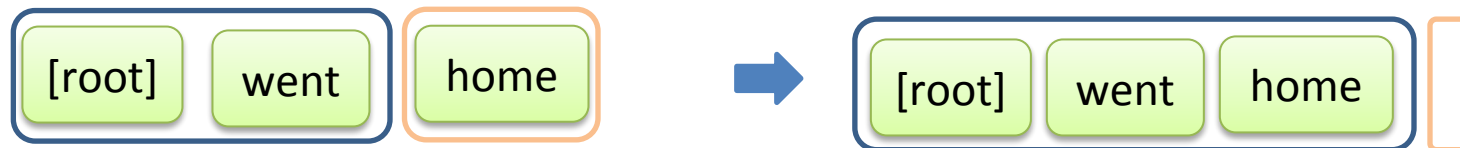


Analysis of “I went home”

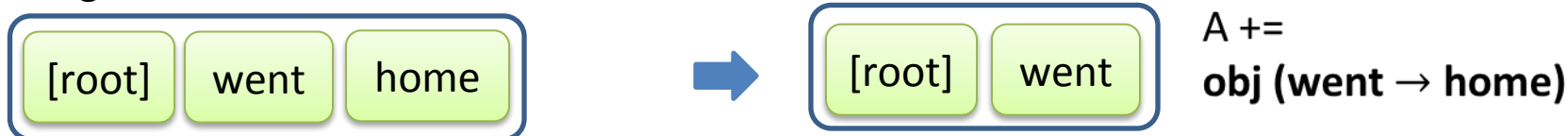
Left Arc



Shift



Right Arc



Right Arc



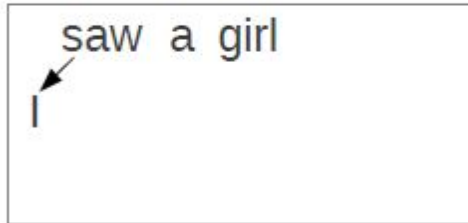
Classification for Shift-Reduce

- Given a **state**:

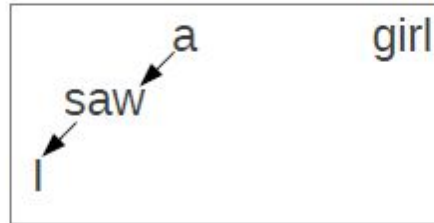


- Which **action** do we choose?

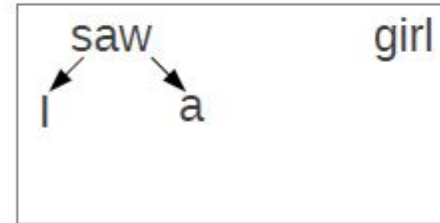
shift ?



r left ?



r right ?



- Correct actions → correct tree

Shift-Reduce as a supervised classification task

- Given the current state ($\langle \text{stack}, \text{buffer}, A \rangle$) predict the next action.
- We have a weight vector for “shift” “left arc” “right arc”

$$w_s \ w_l \ w_r$$

- Calculate feature functions from the queue and stack

$$\phi(\text{queue}, \text{stack})$$

Word in stack, POS of word, Word in buffer and POS of Word in buffer

- Multiply the feature functions to get scores

$$s_s = w_s * \phi(\text{queue}, \text{stack})$$

- Take the highest score to predict the next transition
- Quite fast! $O(N)$ in length of sentence.

Three Problems

- To apply ML in situations like this we have three problems
 1. Discovering features that are useful indicators of what to do in any situation
 2. Acquiring the necessary training data
 - Treebanks associate sentences with their corresponding trees
 - We need **parser states** paired with their corresponding correct **operators**
 3. Training a classifier

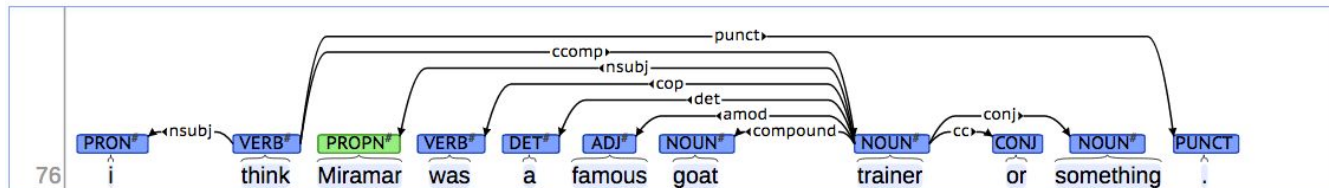
Preparing the Training Set

- Get dependency parses from treebank.
- Compute correct sequence of “oracle” shift-reduce parse actions (transitions, t_i) at each step from gold-standard parse trees.
- Determine correct parse sequence by using a “shortest stack” oracle which always prefers LeftArc over Shift.
- At each stage the oracle chooses
 1. **Left** if the relation to be added is in the reference tree.
 2. **Right** if the resulting relation is in the correct tree AND if all the other outgoing relations associated with the word are already in the relation list.
 3. Otherwise **shift**

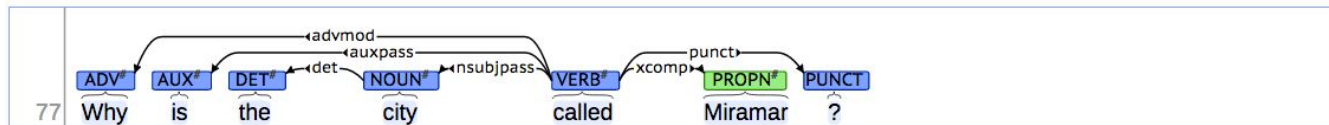
Universal Dependencies treebanks

[Universal Dependencies: <http://universaldependencies.org/>]

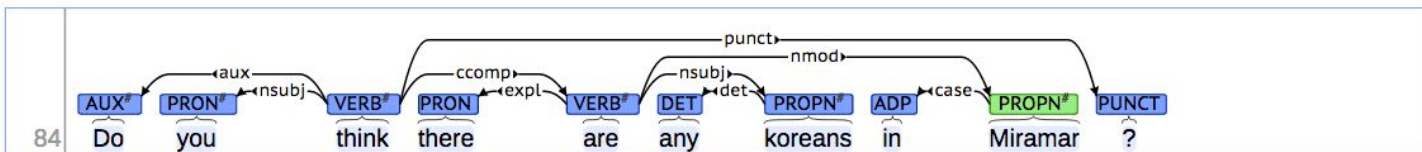
[context] [conllu]



[context] [conllu]



[context] [conllu]



Evaluation

- If we have a test set from a treebank and if we represent parses as a list of relations

(booked, I) (booked, flight) (flight, a) (flight, morning)

- Unlabeled attachment score (UAS) is just what fraction of words were assigned the right head.
- Labeled attachment score (LAS) is what fraction of words were assigned the right head with the right relation.

Labeled Relations

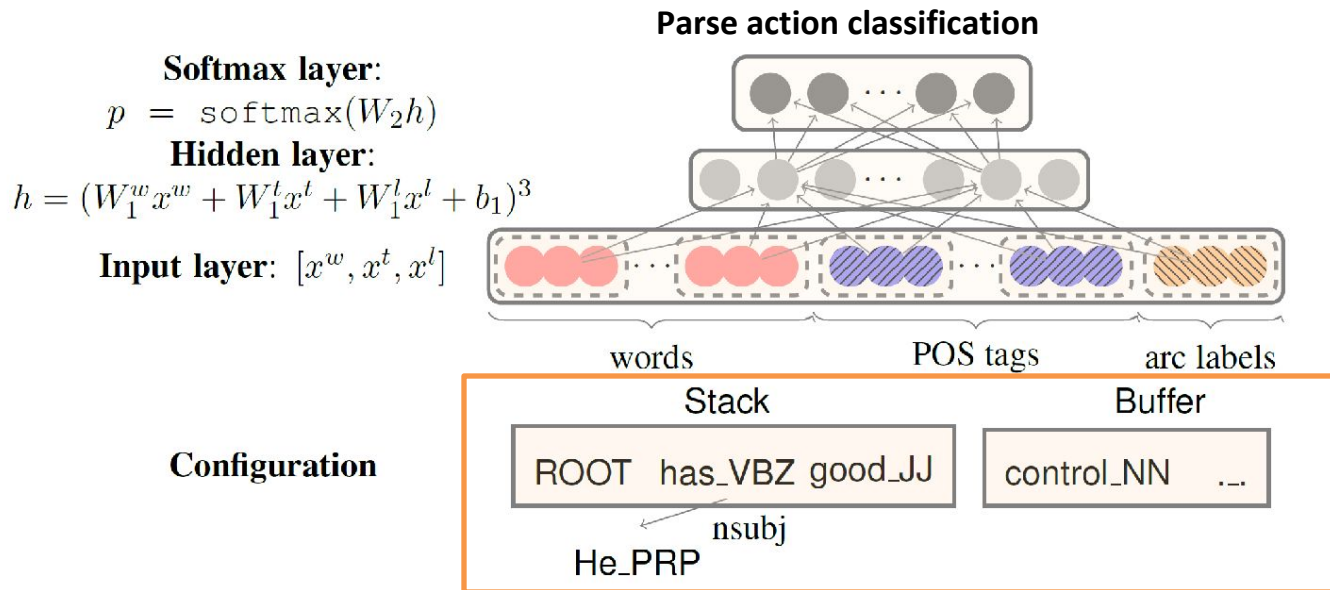
- We really want labeled relations
 - That is, we want things like subject, direct object, indirect object, etc. as relations
- We can add new transitions
 - Replace Left and Right with $\{\text{Left}, \text{Right}\} \times \{\text{all the relations of interest}\}$

Stanford Neural Dependency Parser

(Chen and Manning, 2014)

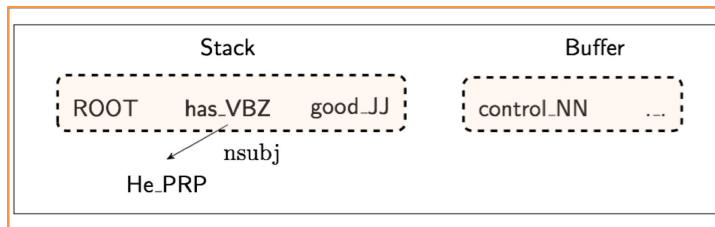
- Train a neural net to choose the best shift-reduce parser action to take at each step.
- Uses features (words, POS tags, arc labels) extracted from the current stack, buffer, and arcs as context.

Neural Architecture



State Representation

Extract a set of tokens from Stack/Buffer



Instead of using one-hot input encodings, words and POS tags are “embedded” in a 50 dimensional set of input features.

	Word	POS	dep
S1	good	JJ	0
S2	has	VBZ	0
B1	control	NN	0
lc(S1)	0	0	0
rc(S1)	0	0	0
lc(S2)	He	PRP	nsubj
rc(S2)	0	0	0

Embeddings express similarities
POS: NN similar to NNS
deps: amod similar num

Concatenate their vector embeddings