Assignment Project

Tra

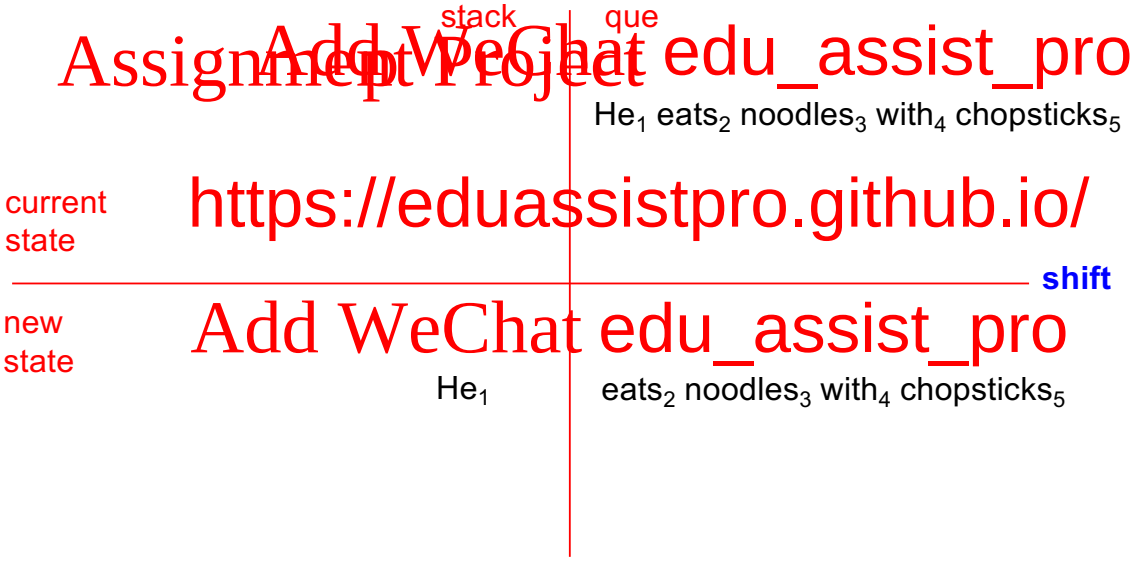
Transition-based syntactic parsing

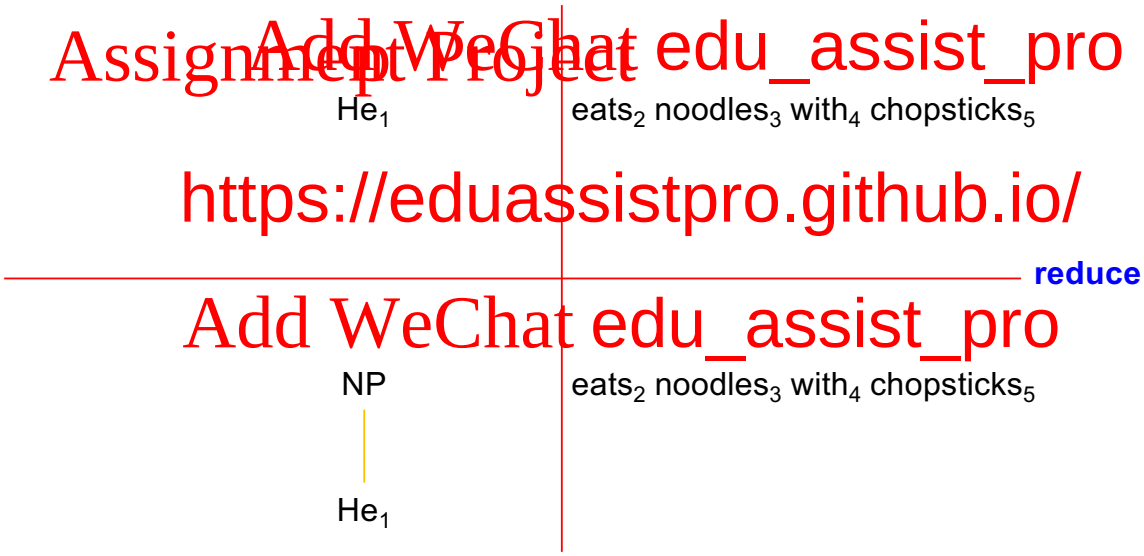
- Transition-based constituent parsing
- Transition-
- Transition-based AMR parsing

Transition-based Constituent Parsing

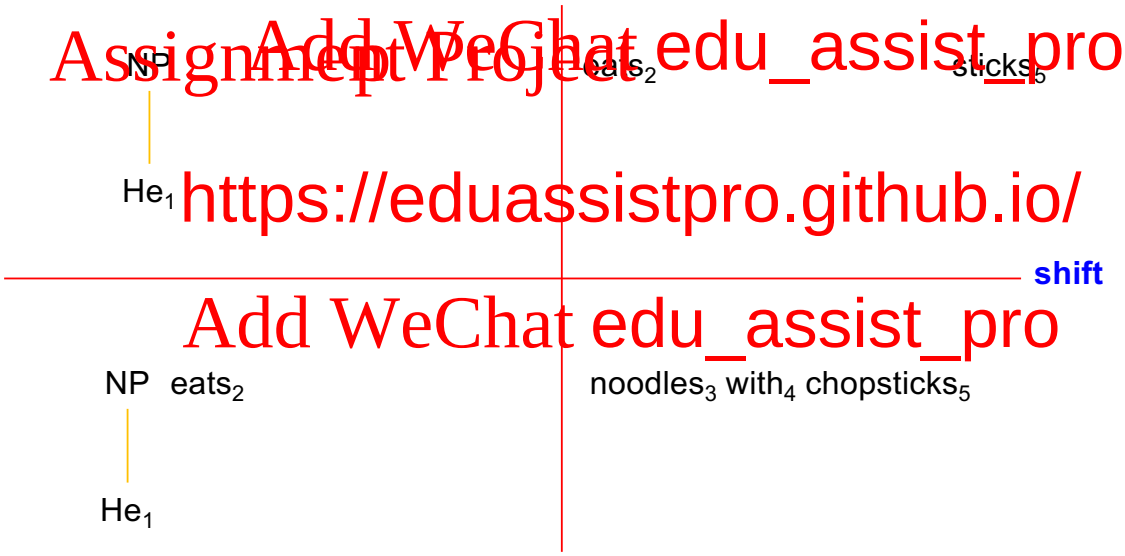
- A transition-based constituent parsing model is a quadruple $C = (S, T, s_0, S_t)$ where:
  - S is a set of parser *states* or *con*
  - T is a set of actions, e.g., *shift*,
  - s_0 is an in ce into a un
  - $S_t \in S$
- An action $t \in T$ is a transition funct s the current state into a new state
- A state $s \in S$ is defined as a tuple $s = (\alpha, \beta)$ where α is a *stack* that holds already constructed subtrees, and β is a *queue* which is used to store words that is yet to be processed.

Shift-Reduce





Shift-Red gorithm



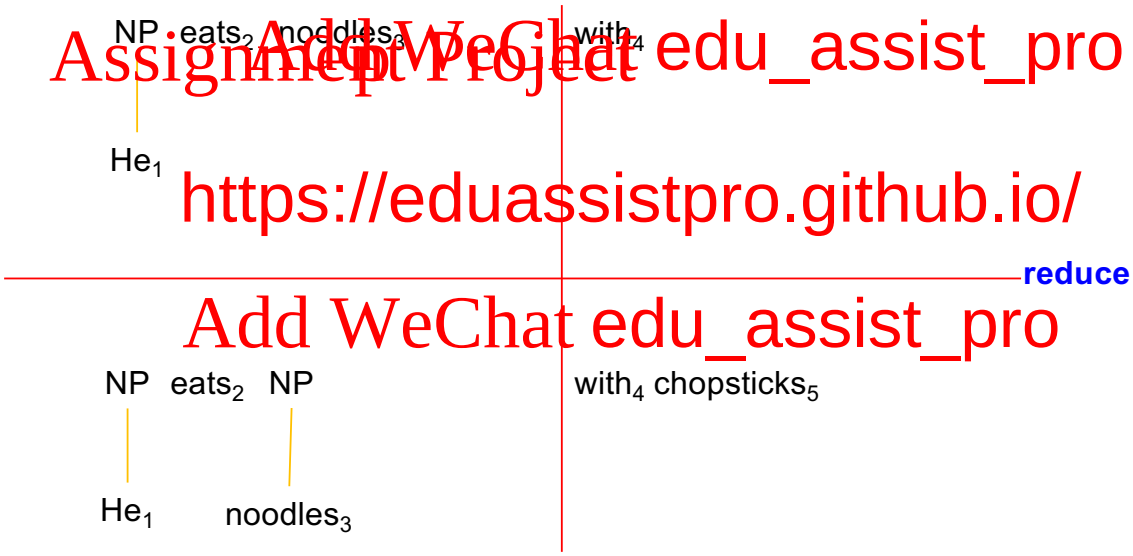
Shift-Redu

orithm

Shift-Redu https://eduassistpro.github.io/ orithm

Assignment Project Exam Help

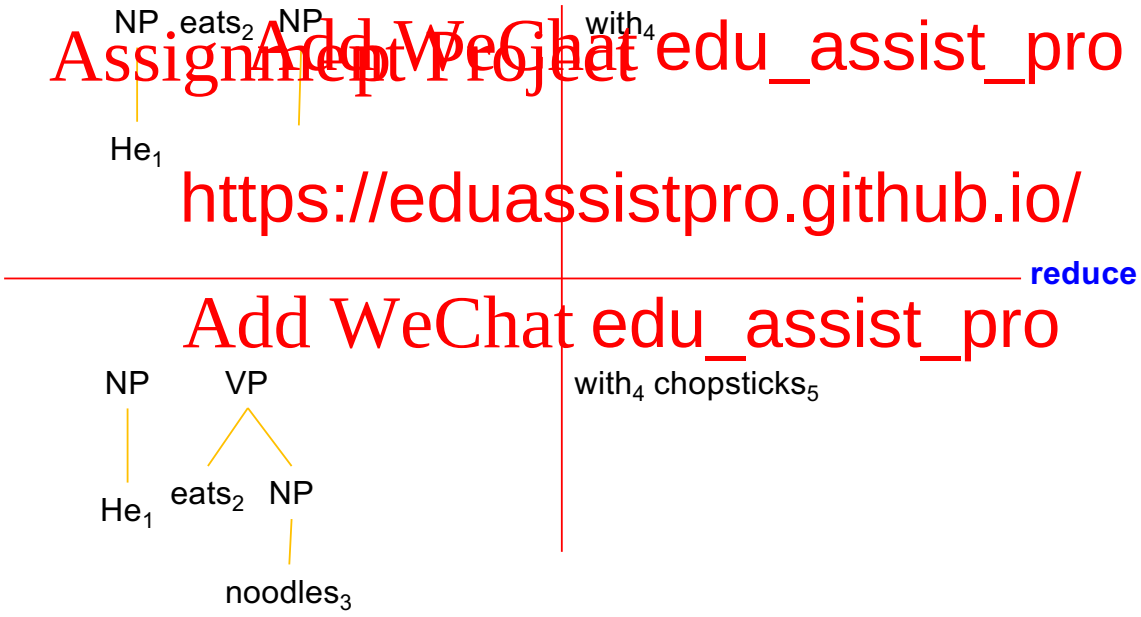
Assignment Project Add WeChat edu_assist_pro



https://eduassistpro.github.io/

Add WeChat edu_assist_pro

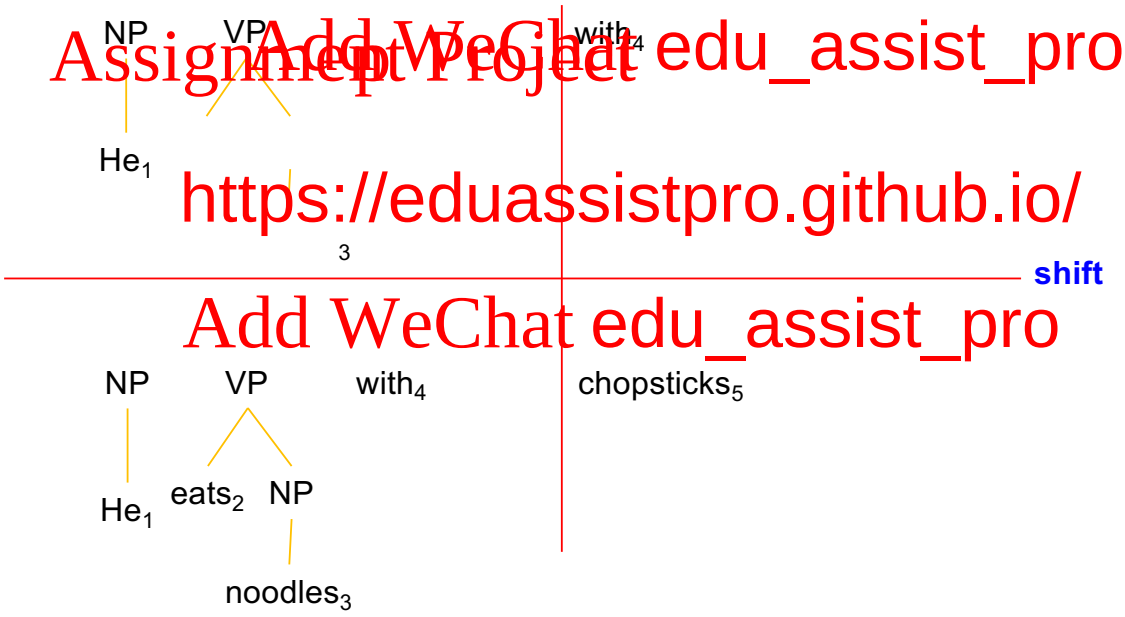
Shift-Redu https://eduassistpro.github.io/ orithm



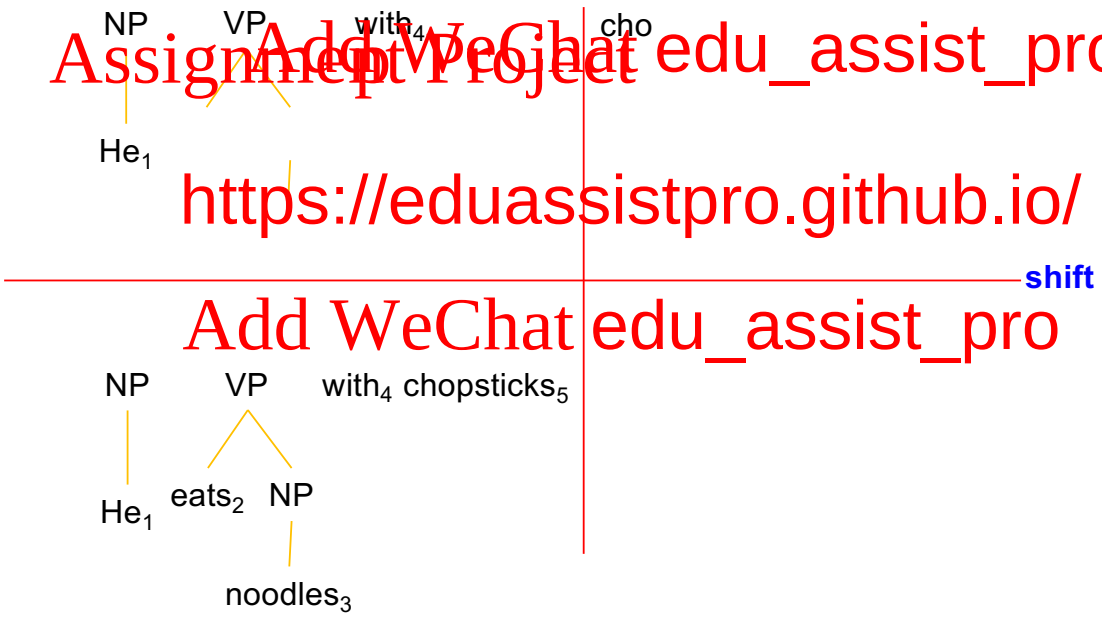
Shift-Redu https://eduassistpro.github.io/ orithm

Assignment Project Exam Help

Assignment Project Add WeChat edu_assist_pro

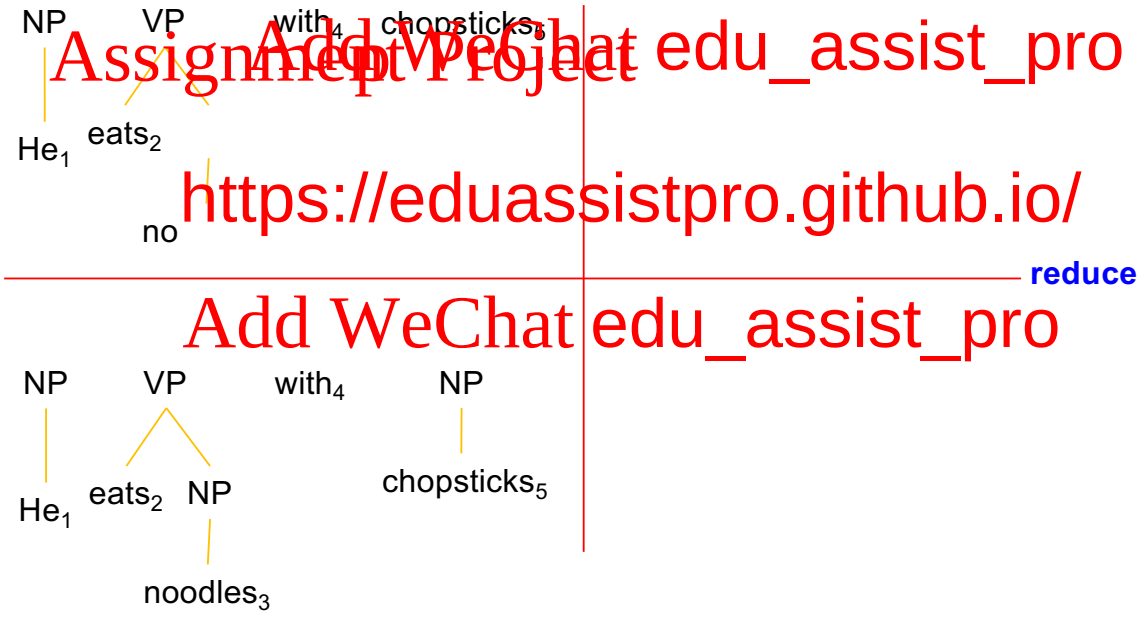


Shift-Redu orithm



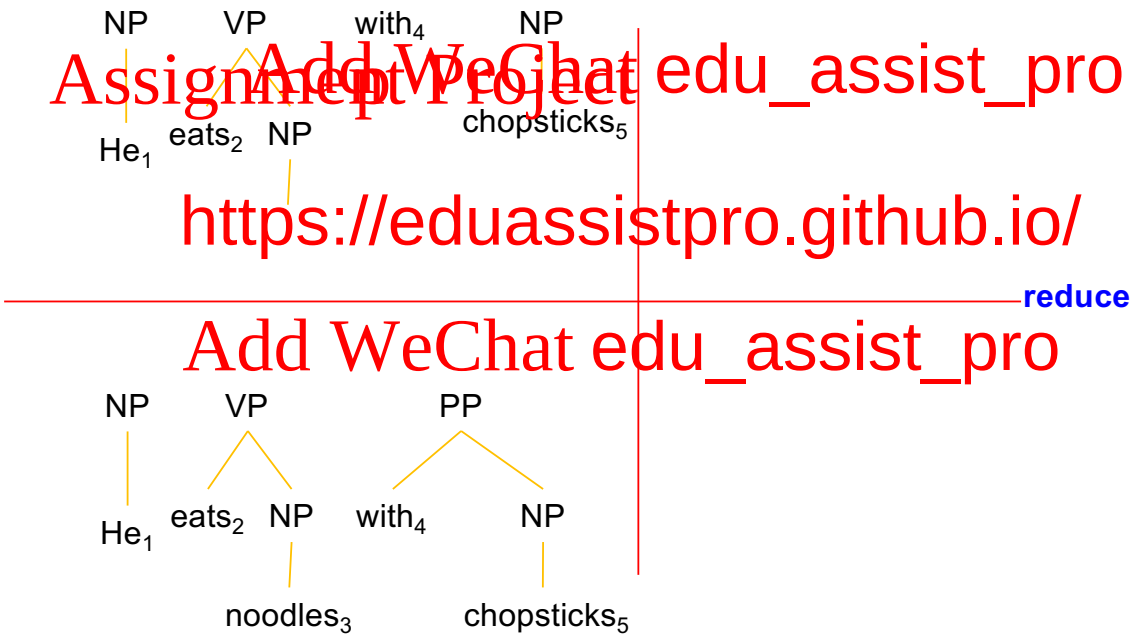
Shift-Redu

orithm



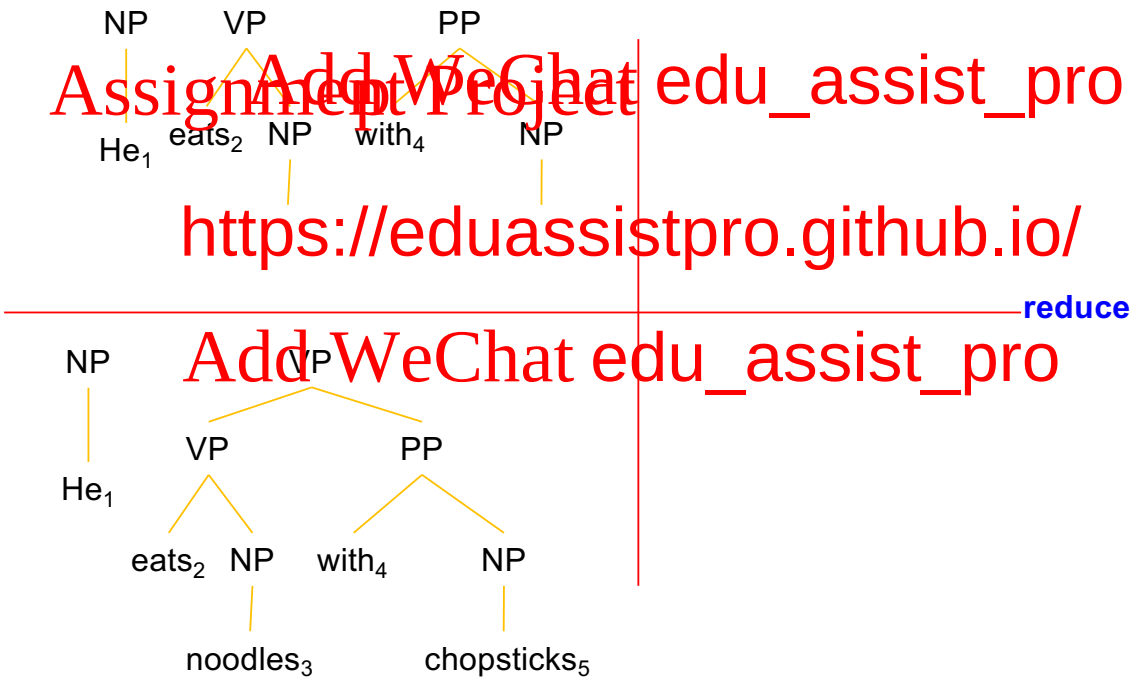
Shift-Redu

orithm

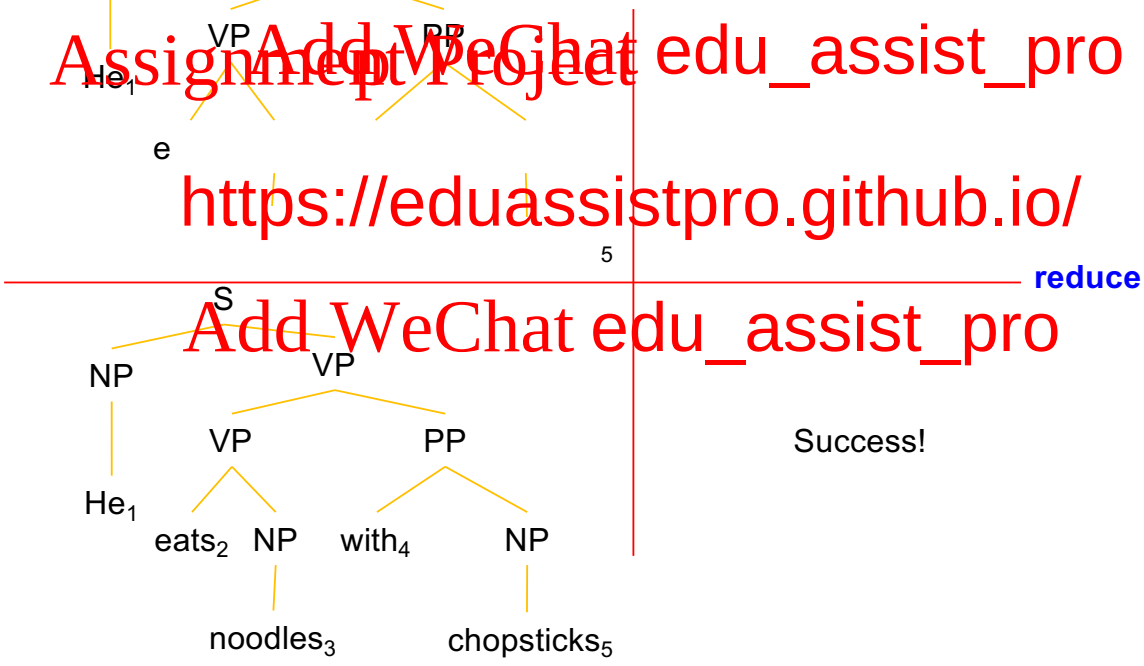


Shift-Redu

crithm



Shift-Redu rithm



“Oracle”

- The oracle is a sequence of actions that lead to th parse of a sentence.
- When trainin t map a gold parse tre
- We can learn a model by comparing the ora action sequences and update the param

The Perceptron learning algorithm

1: ***Input***: Training e es (x_i, y_i)
2: ***Initialization***: Set $\boldsymbol{\theta} = \mathbf{0}$
3: **for** $t \leftarrow 1$,
4: **for** $i \leftarrow$
5: $z_i \leftarrow \underset{z \in GEN(x_i)}{}$
6: **if** $z_i \neq y_i$ **then**
7: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \mathbf{f}(x_i, y_i) - \mathbf{f}$
8: ***Output***: Parameters $\boldsymbol{\theta}$

Lexcialized transition-based parsing actions

- Each action t s a state into a new state.
  - SHIFT(SH): remove the first word-POS pair from β, and pushes it onto the top of σ;
  - REDUCE-UNARY-X(RU-X): pop the to σ construct a new binary node labeled with ubtree, and then . The head of the new sub
  - REDU o subtrees from σ, combine them into a new tree w with X, then push the new subtree bac σ. The left (L) and right (R) versions of the action indicate whether the head of the new subtree is inherited from its left or right child.
- A parsing state $s \in S$ is defined as a tuple $s = (\sigma, \beta)$, where σ is a stack that is maintained to hold the partial parsing structures that are already constructed and β is a queue used to store unprocessed input (typically word-POS tag pairs).

Updating feature weights

$$\nabla_\theta \left(\begin{bmatrix} p_0tc = N - NP \sim shift & 1 \\ p_0tc = N - NP \sim reduce & 0 \\ p_0wc = noodles \quad NP \quad shift & 0 \\ p_0wc = p_1t & 1 \\ p_1tc = V - V \sim reduce & 1 \\ p_1wc = eats - V \sim shift & 0 \\ p_1wc = eats - V \sim reduce & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} \right) = \begin{bmatrix} -1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Notes: The feature $p_0tc = N - NP$ predicts a "shift" action when the oracle action should be "reduce".

Transition-based parsing features

Type	Feature Templates
unigrams	$p_0 wc$, $p_3 t_0$, $p_3 wc$, $q_0 wt$, $q_1 wt$, $q_2 wt$, $q_3 wt$, $p_{0l} wc$, $p_{0r} wc$, $p_{0u} wc$, $p_{1l} wc$, p_{1r}
bigrams	$p_0 wp_1 w$, $p_0 wp_1 c$, $p_0 cp_1 c$, $p_0 cq_0 t$, $p_0 wq_0 t$, $p_0 wq_0 w$, $p_1 wq_0 w$, $p_1 w$, $cq_0 t$, $p_0 cq_0 w$
trigrams	$p_0 cp_1 cp_2 c$, $p_0 cp_1 cp_2 w$, $p_0 cp_1 wq_0 t$, $p_0 cp_1 cq_0 w$

Baseline features, where p_i represents the i_{th} subtree in the stack σ and q_i denotes the i_{th} item in the queue β. w refers to the head word, t refers to the head POS, and c refers to the constituent label. p_{il} and p_{ir} refer to the left and right child for a binary subtree p_i, and p_{iu} refers to the child of a unary subtree p_i.

Feature vector

feature	count	feature	count
$p_0 tc$=N-NP˜shift	0	$p_0 tc$=N-NP˜reduce	1
$p_0 wc$=noodles-NP˜shift	0	p_0 P˜reduce	1
$p_1 tc$=V-V˜shift	0	p_1 e	1
$p_1 wc$=eats-V˜		-V˜reduce	1
$p_{0u} wc$=noodle		dles-N˜reduce	1
$q_0 wt$=with-P˜shift	0	reduce	1
$q_1 wt$=chopsticks-N˜shift	0	cks-N˜reduce	1
...	...		...

Notes: Feature count for one configuration. The total count for a sentence will be a sum over all configurations in the derivation of the syntactic structure of the sentence

Beam Search

Input: A POS-tagged s
Output: A constituent parse tree

1: $beam_0 \leftarrow \{s_0\}$ ▷ initialization
2: $i \leftarrow 0$ ▷ step index
3: **loop**
4: $P \leftarrow \{\}$ priority queue
5: **while** *bea*
6: $s \leftarrow$
7: **for** all p
8: $s_{new} \leftarrow$ apply t to s
9: score s_{new}
10: insert s_{new} into P
11: $beam_{i+1} \leftarrow k$ best states of P
12: $s_{best} \leftarrow$ best state in $beam_{i+1}$
13: **if** $s_{best} \in S_t$ **then**
14: **return** s_{best}
15: $i \leftarrow i + 1$

CFG based parsing vs transition-based parsing

- A transition-based parser scores the actions while a PCFG based parsing model scores the rules
- It's customary to use the beam search algorit transition-b
- The transitio dependency parsing as well as graph-ba g
- Learning for transition-based parsing basically any type of classifier, including neural network models

Constituent parsing as sequence-to-sequence learning

- When you have a hammer, everything looks li
  we apply sequence-to-sequence modelin g?
- Our input is the r
  output is a cons
  sequence?
- We need to linearize the tree in such a way that it is fully reversible and can be mapped back to a tree.

Tree linearization

(S (NP NNP)NP (VP VBD (NP (NP NNS)NP (PP IN (NP NNS)NP)PP)NP)VP)S

Alternative linearizations possible, e.g., a sequence of shift reduction actions.

Tricks and tips

- Normalizing POS tags: replacing all POS tags with XX to further reduce the size of the output vocabulary
- Input inversing: Inversing the input sentenc
  outputs trees helps (similar effects have been s
  inversing the s
- There is no gua
  tree structur
  will be well-formed, but you need to do som
  or checking to make sure the trees are well-forme
  matching parentheses, etc.
- In practical systems, LSTM-RNNs are used rather than vanilla RNNs, which are easier to explain but don't work as well in practice

BLEU: a metric for measuring similarity of sequences

- **BLEU** stands for Bilingual Evaluation Understudy, and it is the most popular MT evaluation metric.
- N-gram overlap between machine transla reference tra
- Compute pre
- Add brevity penalty (for too short transla

$$\text{BLEU} = \min\left(1, \exp\left(1 - \frac{\textit{reference-length}}{\textit{output-length}}\right)\right) \left(\prod_{i=1}^{4} \textit{precision}_i\right)^{\frac{1}{4}}$$

- In logarithms: $\exp \frac{1}{N} \sum_{n=1}^{N} \log P_n$
- Typically computed over the entire corpus, not single sentences

Example

Metric	System	
precision (1gram)	3/6	
prec		/5
prec		/4
precision (4gram)	0	
brevity penalty	6	
BLEU	0	

Fine prints of BLEU

- To avoid computing $\log 0$, all precisions are smoothed to ensure that they are positive.
- Each n-gram in the reference can be used at m
  that *to to to to t* $_1 = 1$ against the reference *t*
- **Brevity penality** is applied to transl ter than the reference.
- Normalization often performed on reference and hypothesis translations to get better match.