A Fast and Accurate Dependency Parser using Neural Networks

Danqi Chen & Christopher D. Manning

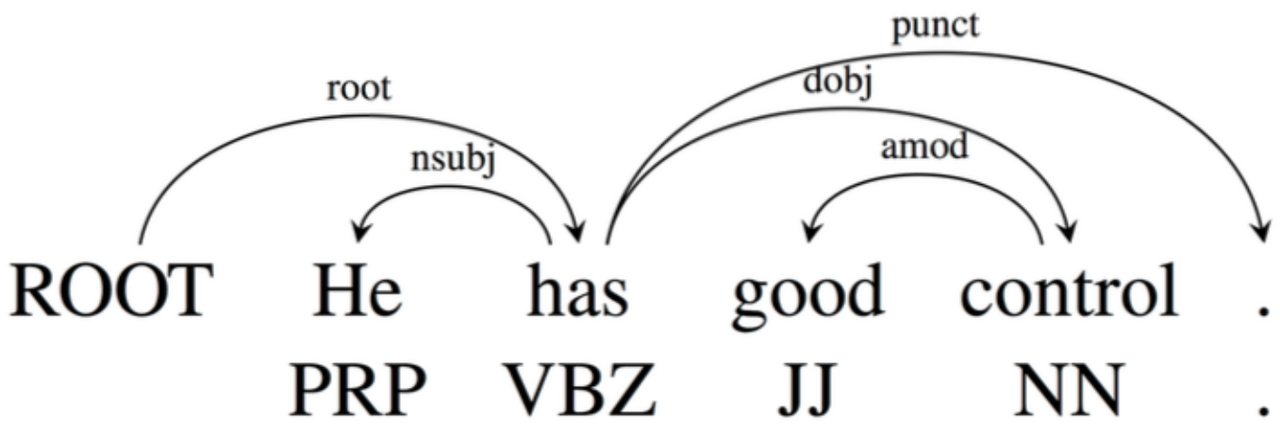
Qiming Chen
qc2195
Apr. 8. 2015

Dependency Parsing

- Parsing: He has good control .

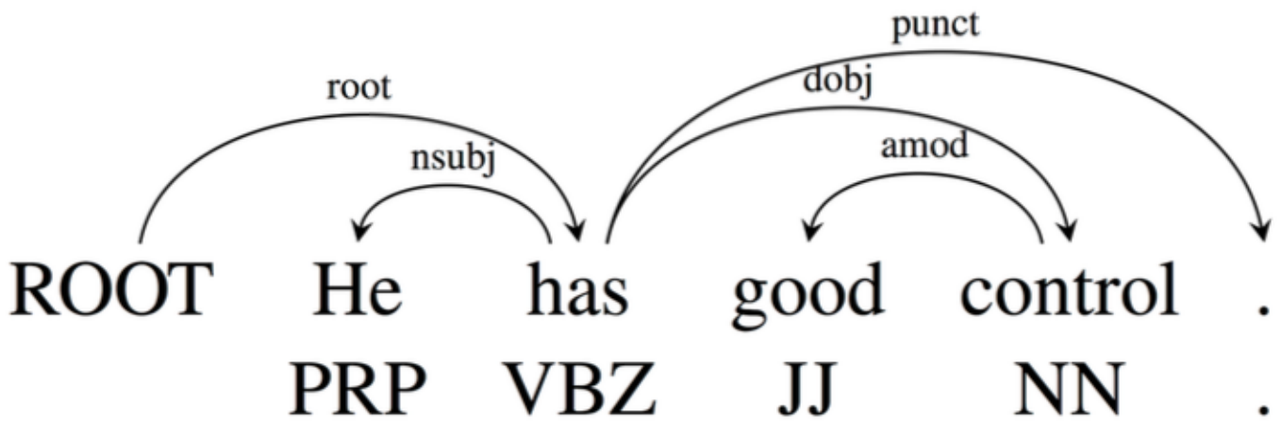
Dependency Parsing

- Parsing: He has good control .



Dependency Parsing

- Parsing: He has good control .



- Goal: accurate and fast parsing

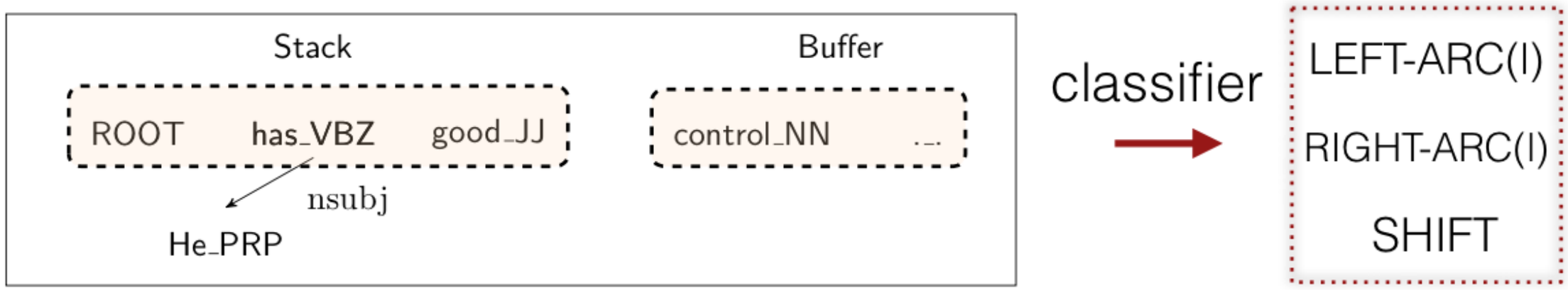
Transition-based Parsing

- A configuration = a stack, a buffer and some dependency arcs
- arc-standard system is employed

Transition-based Parsing

- A configuration = a stack, a buffer and some dependency arcs

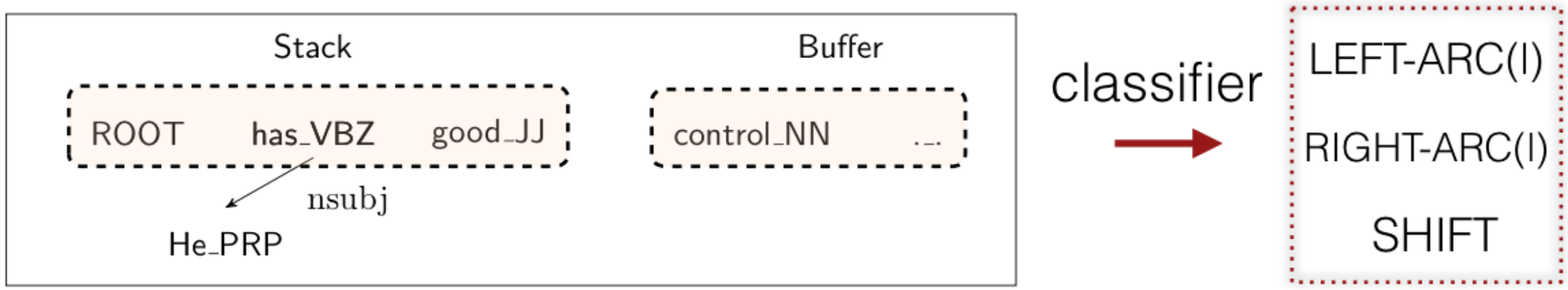
- arc-standard system is employed



Transition-based Parsing

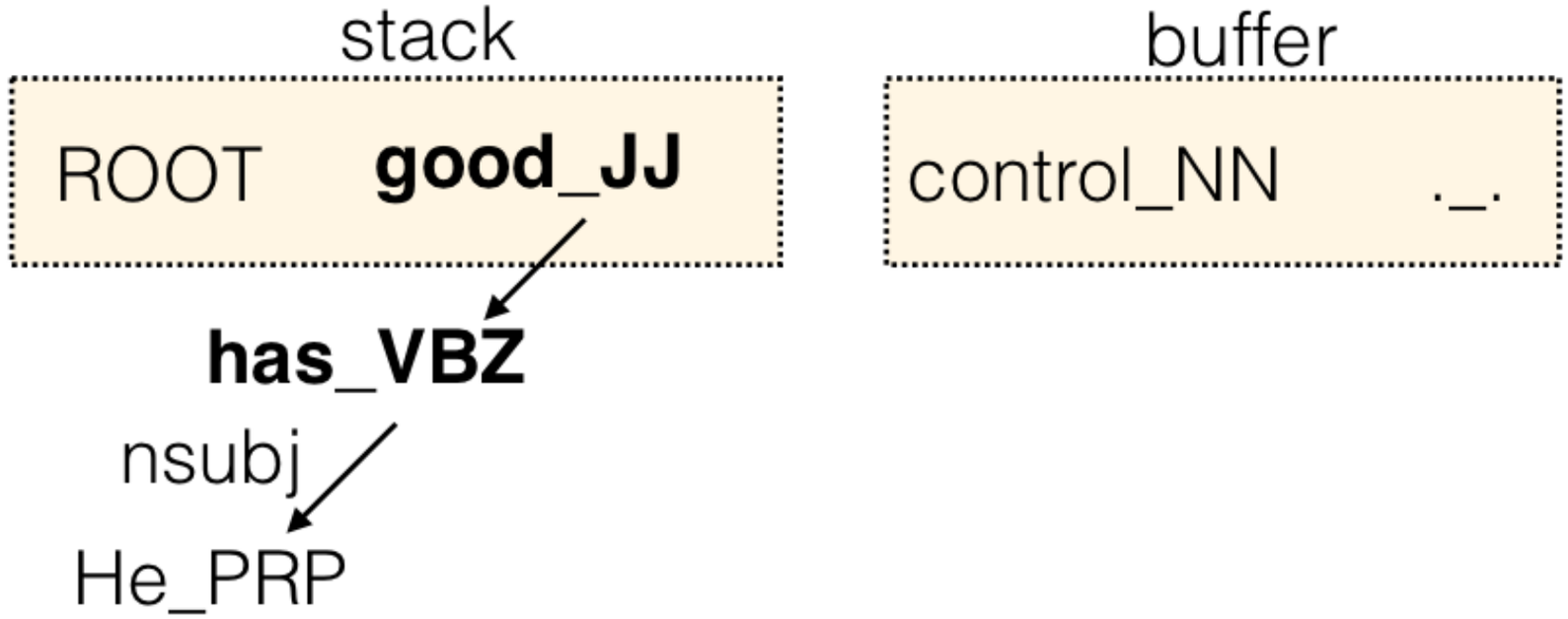
- A configuration = a stack, a buffer and some dependency arcs

- arc-standard system is employed



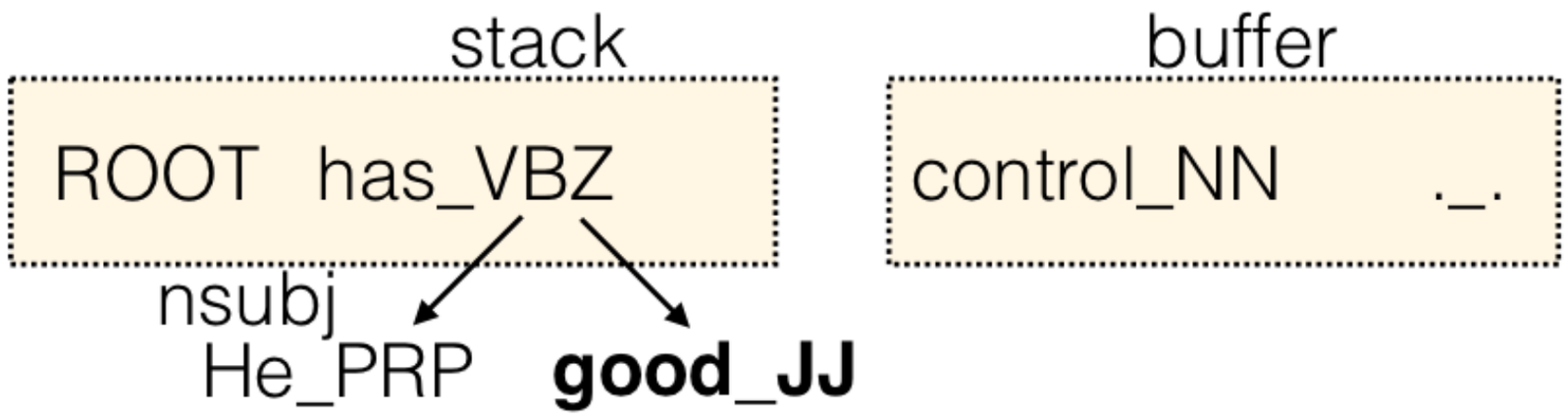
LEFT-ARC


stack
buffer
ROOT has_VBZ good_JJ
control_NN ._.
nsubj
He_PRP

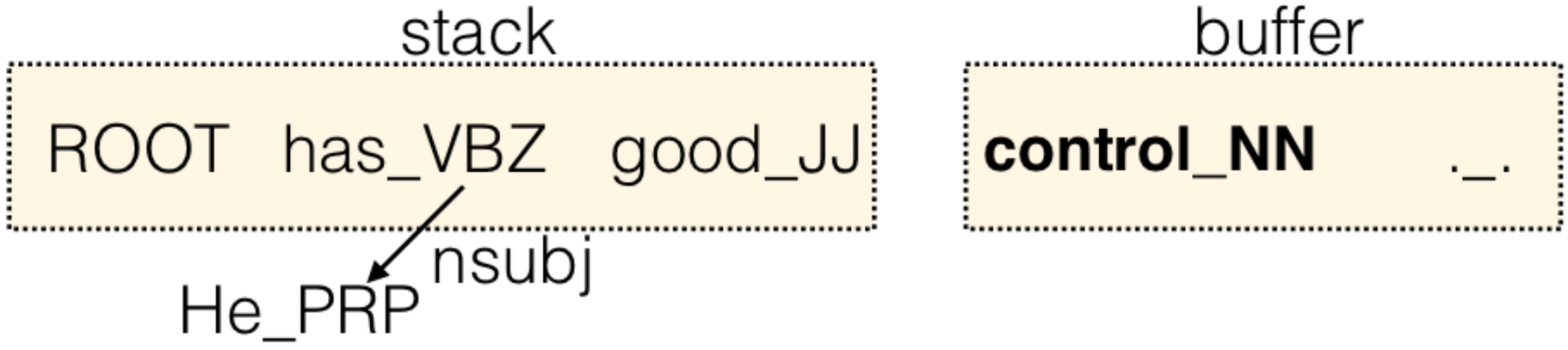

stack
buffer
ROOT good_JJ
control_NN ._.
has_VBZ
nsubj
He_PRP

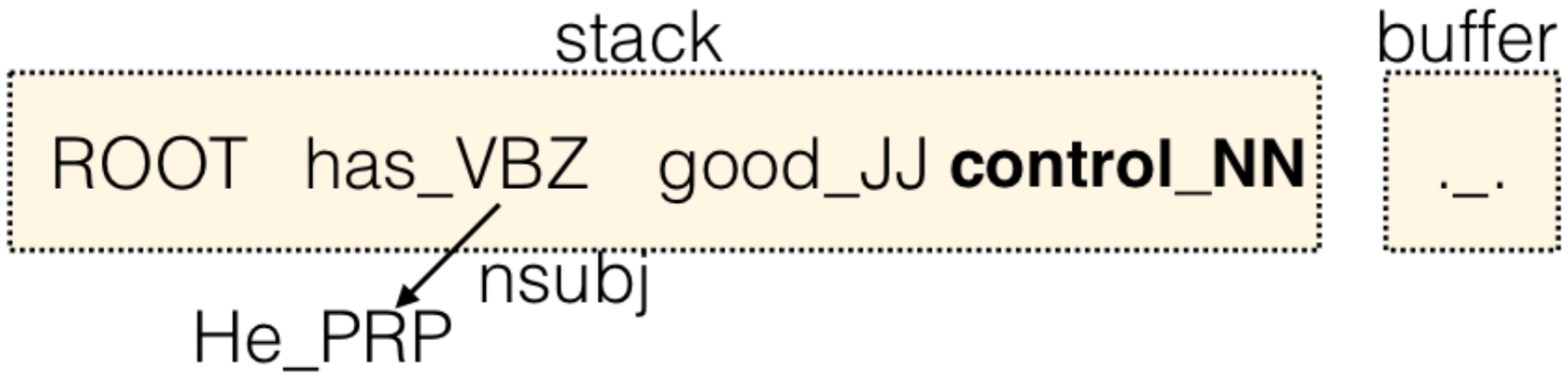
RIGHT-ARC


stack
buffer
ROOT has_VBZ good_JJ
control_NN ._.
nsubj
He_PRP


stack
buffer
ROOT has_VBZ
control_NN ._.
nsubj
He_PRP
good_JJ

SHIFT


stack
buffer
ROOT has_VBZ good_JJ
control_NN ._.
nsubj
He_PRP


stack
buffer
ROOT has_VBZ good_JJ control_NN
._.
nsubj
He_PRP

Traditional Features

binary, sparse
dim =10^6 ~ 10^7

0	0	0	1	0	0	1	0	…	0	0	1	0

Indicator features

$$s_2.w = \text{has} \wedge s_2.t = \text{VBZ}$$
$$s_1.w = \text{good} \wedge s_1.t = \text{JJ} \wedge b_1.w = \text{control}$$
$$lc(s_2).t = \text{PRP} \wedge s_2.t = \text{VBZ} \wedge s_1.t = \text{JJ}$$
$$lc(s_2).w = \text{He} \wedge lc(s_2).l = \text{nsubj} \wedge s_2.w = \text{has}$$

leftmost child

Traditional Features

- Sparse!
- Incomplete
- Computationally expensive

Neural Networks!

- Learn a dense and compact feature representation
- to encode all the available information
- to model high-order features

Dense Feature Representation

- Represent each word as a d-dimensional dense vector.

- Meanwhile, part-of-speech tags and dependency labels are also represented as d-dimensional vectors.

- NNS (plural noun) should be close to NN (singular noun).

Extracting Tokens from Configuration

- We extract a set of tokens based on the positions:
- And get their word, POS, deps

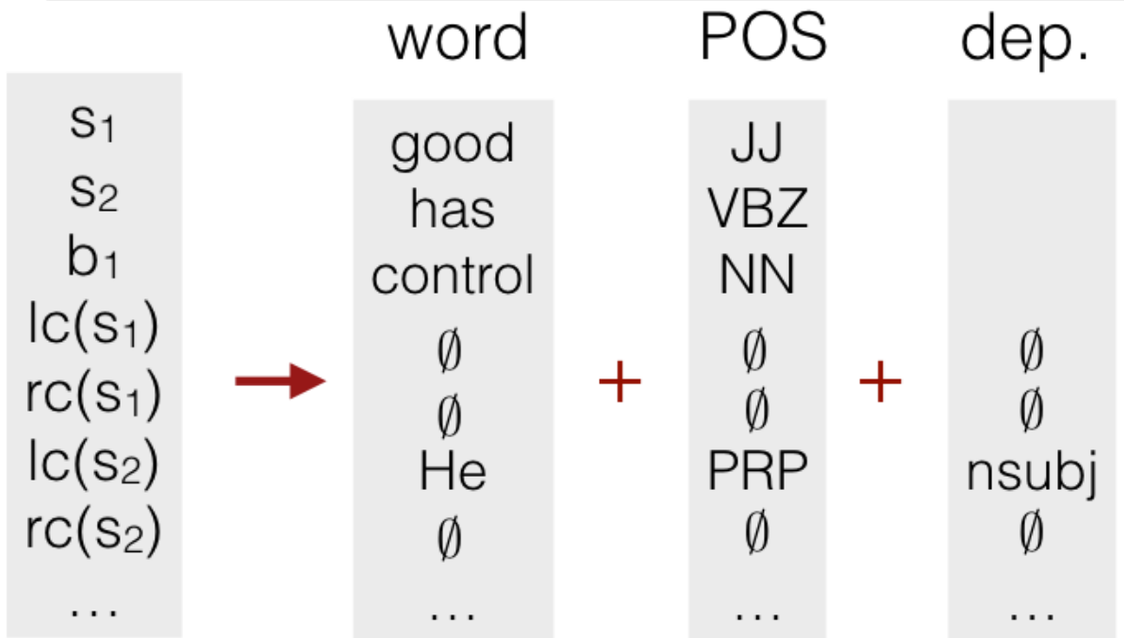
Extracting Tokens from Configuration

- We extract a set of tokens based on the positions:
- And get their word, POS, deps

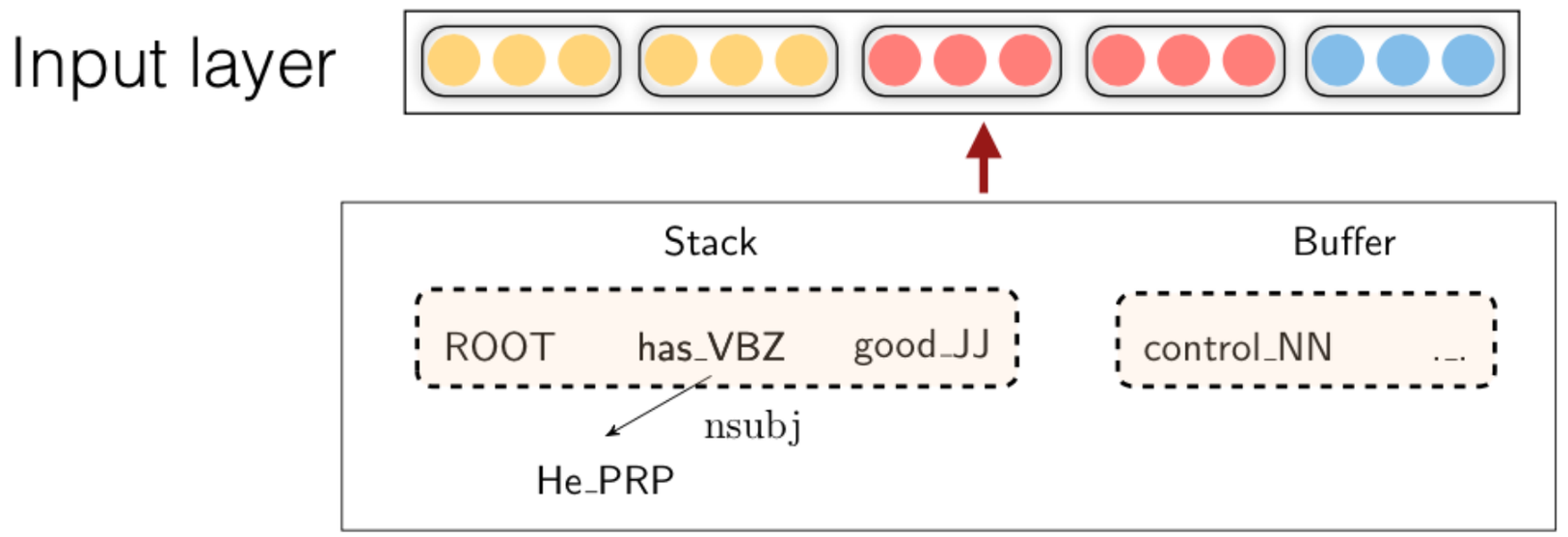
s_1
s_2
b_1
$lc(s_1)$
$rc(s_1)$
$lc(s_2)$
$rc(s_2)$
...

Extracting Tokens from Configuration

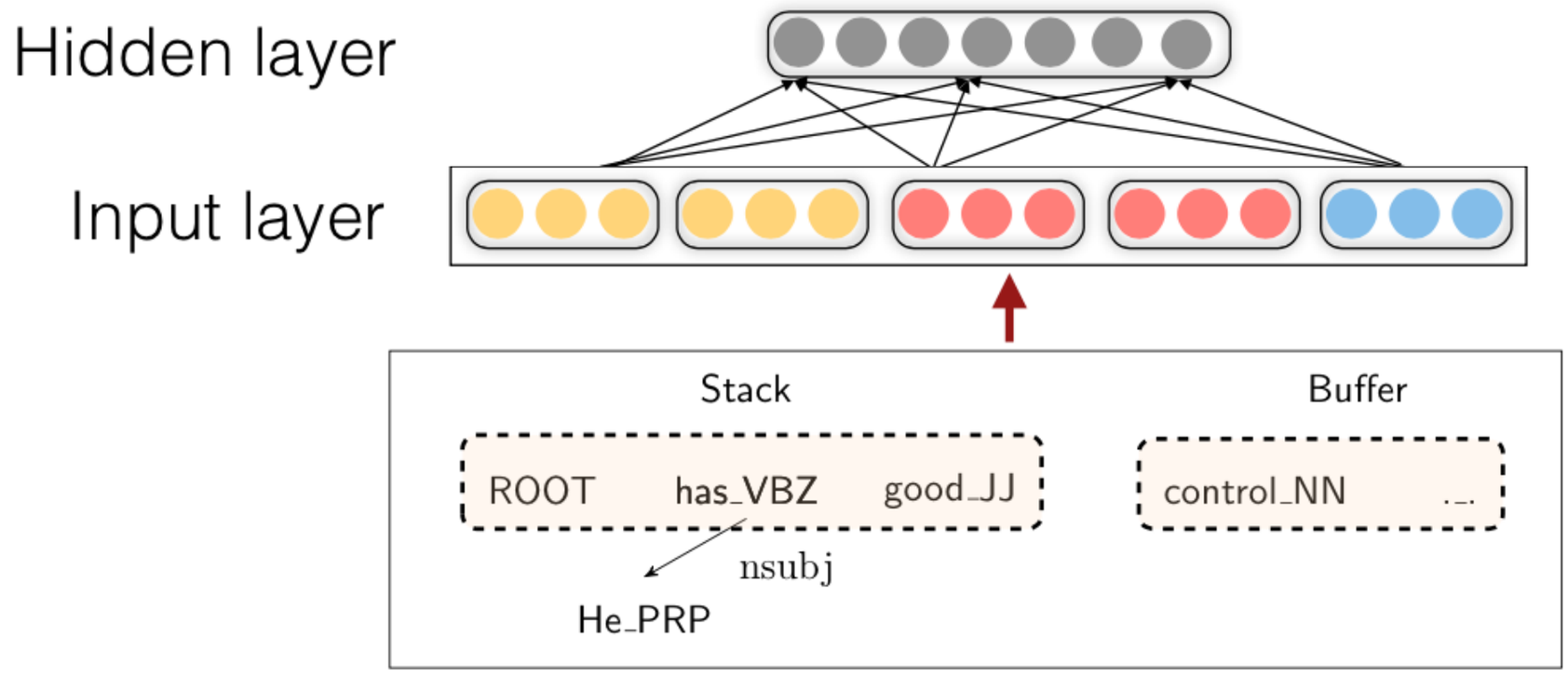
- We extract a set of tokens based on the positions:
- And get their word, POS, deps



Model Architecture

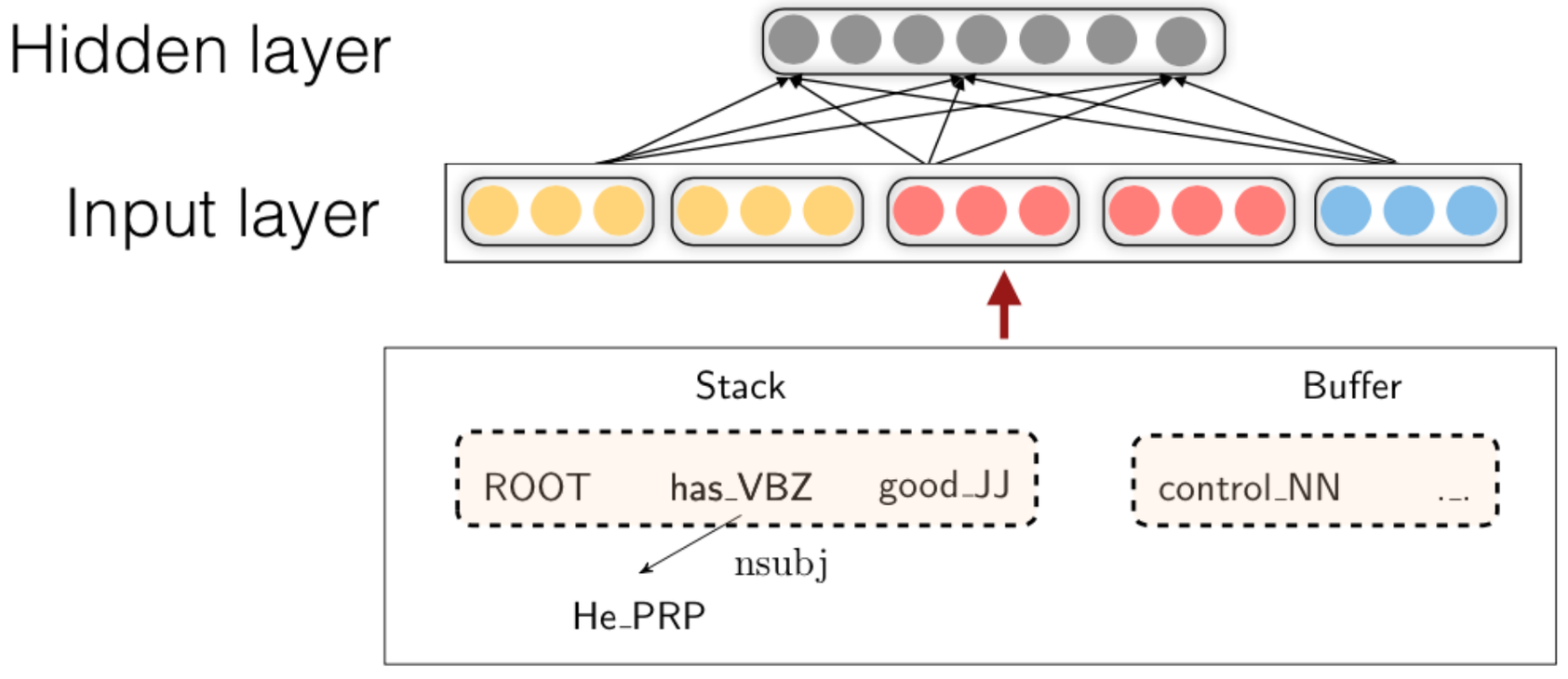


Model Architecture

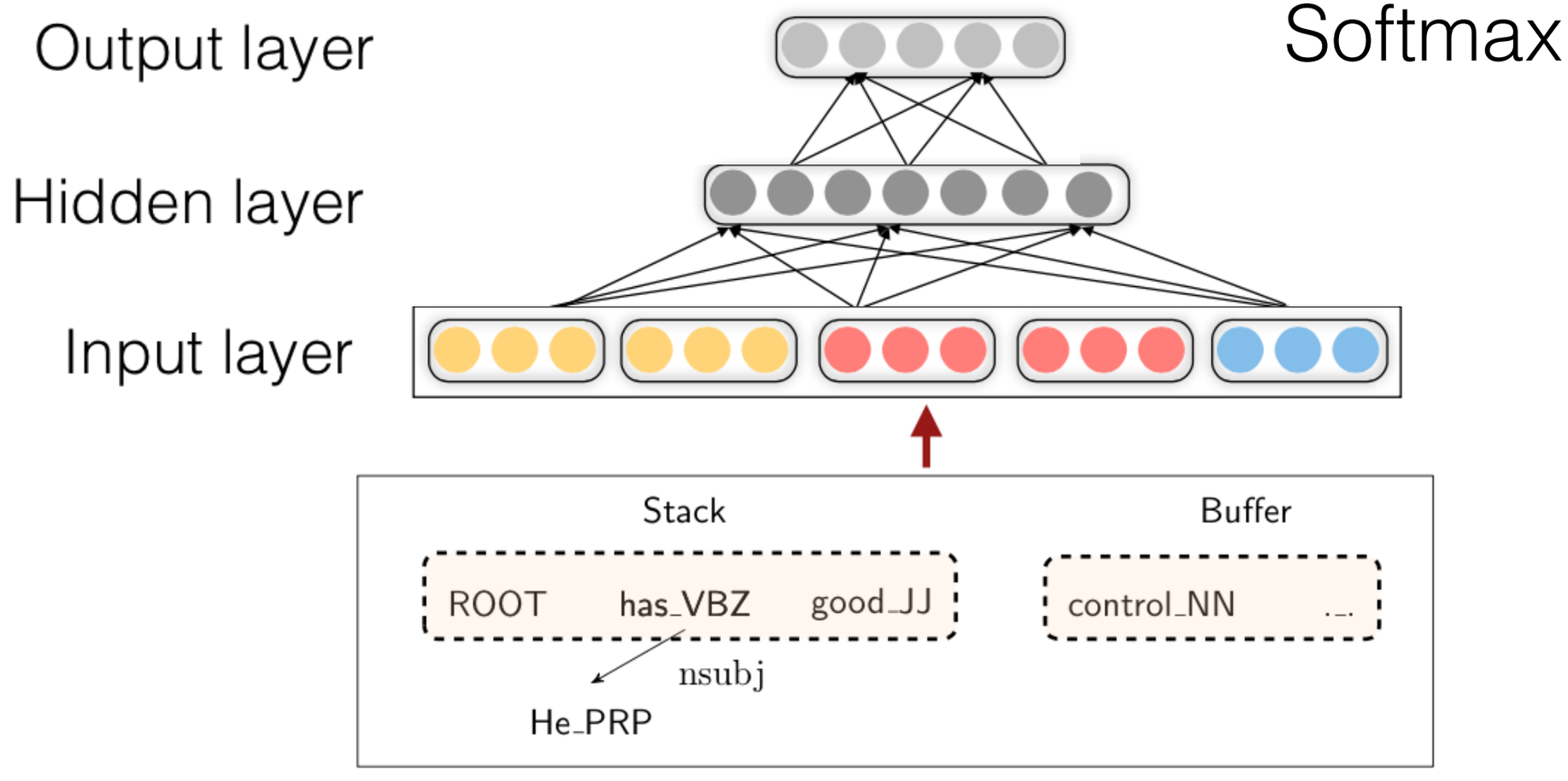


Model Architecture

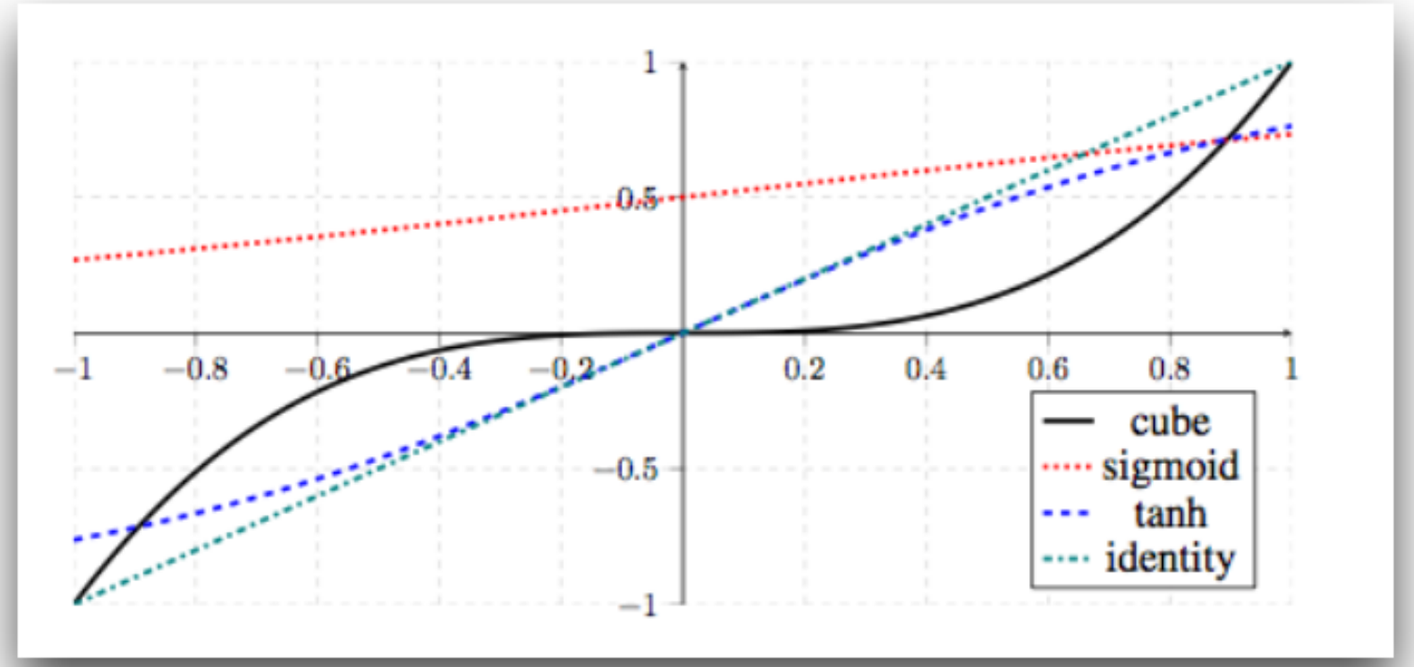
Cube activation function: g(x) = x^3



Model Architecture



Cube Activation Function



$$g(w_1x_1 + \ldots + w_mx_m + b) =$$
$$\sum_{i,j,k}(w_iw_jw_k)x_ix_jx_k + \sum_{i,j} b(w_iw_j)x_ix_j \ldots$$

Better capture the **interaction** terms!

Training

- Data from Penn Tree Bank (Wall Street Journal)
- Generating training examples using a oracle.
- Training objective: cross entropy loss
- Back-propagation to train all embeddings. (Word, POS, dep)
- Initialize word embeddings from pre-trained word vectors

Parsing Speed-up

- Embeddings for popular words, POS tags, dep labels can be pre-computed and cached for speed-up

- 8 ~ 10 times faster.

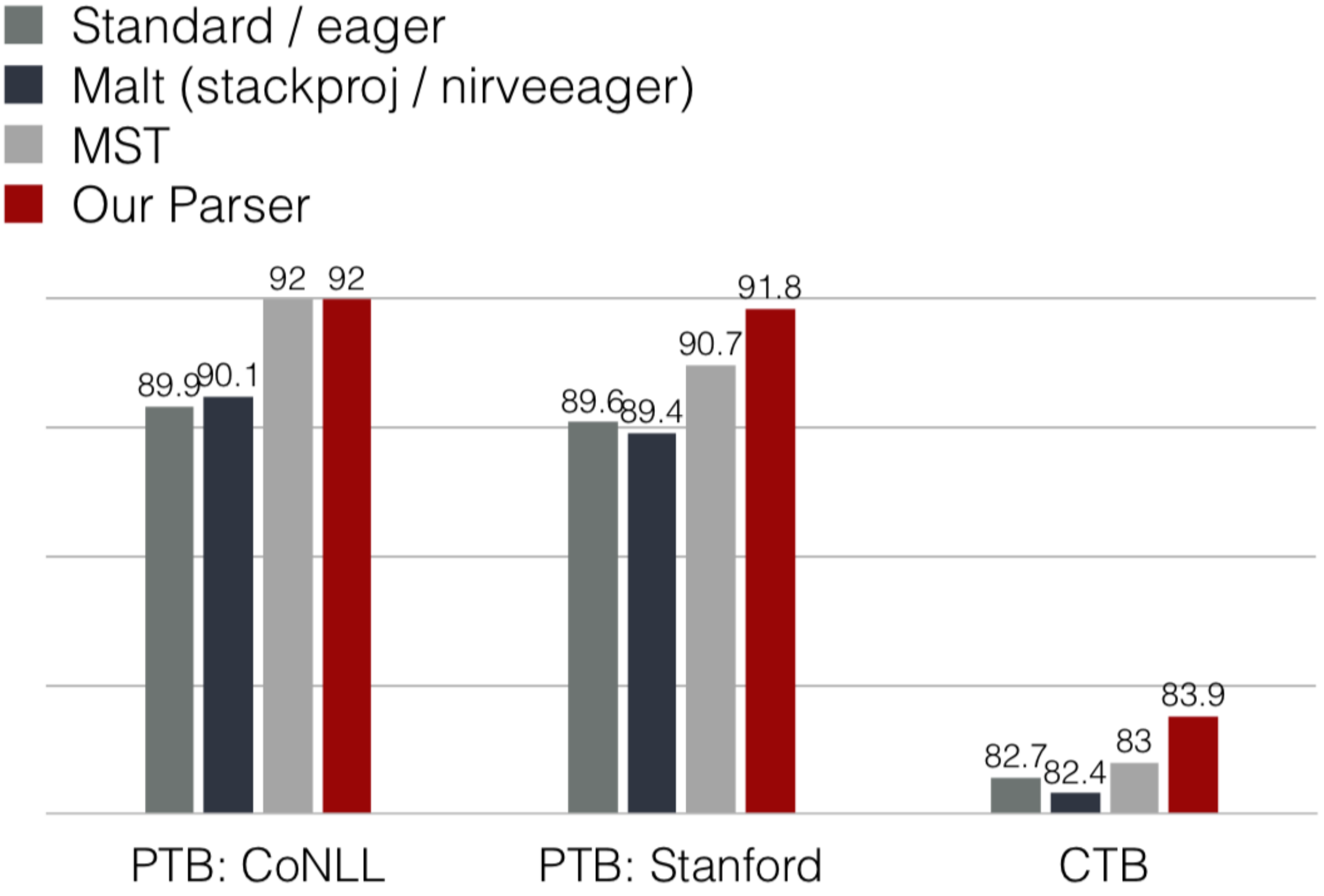
Indicator vs. Dense Features

- Sparse?
- Incomplete?
- Computationally expensive?

Experimental Details

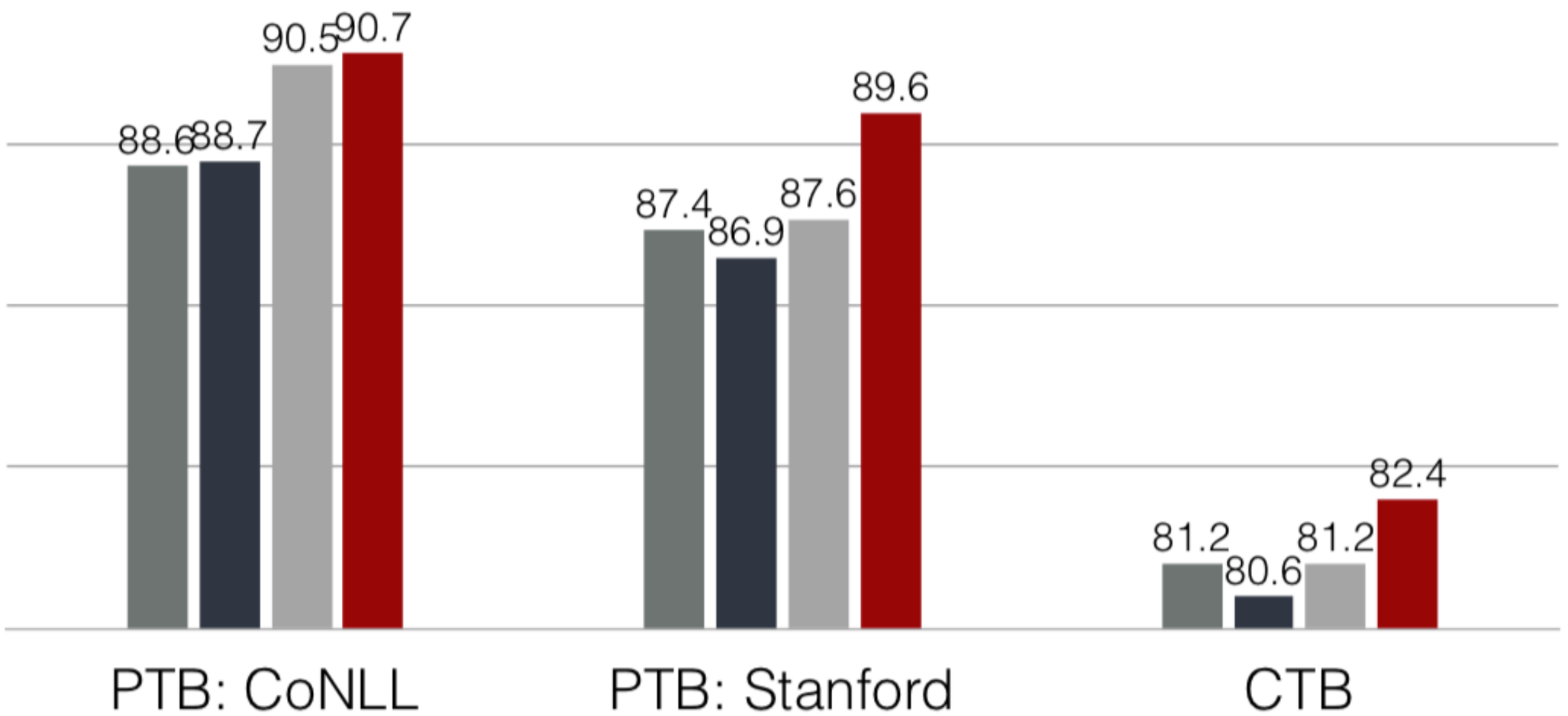
- Embedding size = 50
- Hidden size = 200
- 0.5 dropout on hidden layer
- A rich set of 18 tokens from the configuration
- Pre-trained word embeddings:
  - C & W for English
  - Word2vec for Chinese

Unlabeled Attachment Score (UAS)



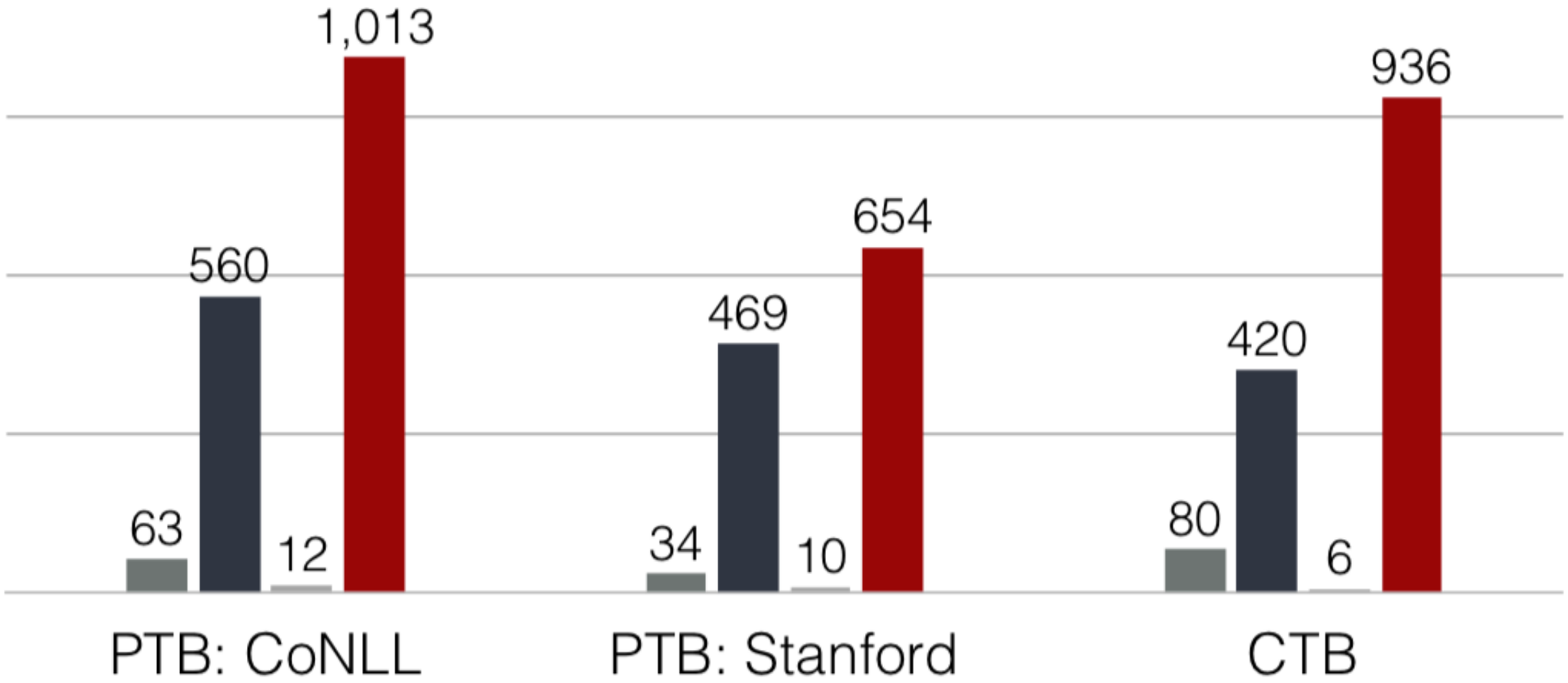
Labeled Attachment Score (LAS)

- Standard / eager
- Malt (stackproj / nirveeager)
- MST
- Our Parser

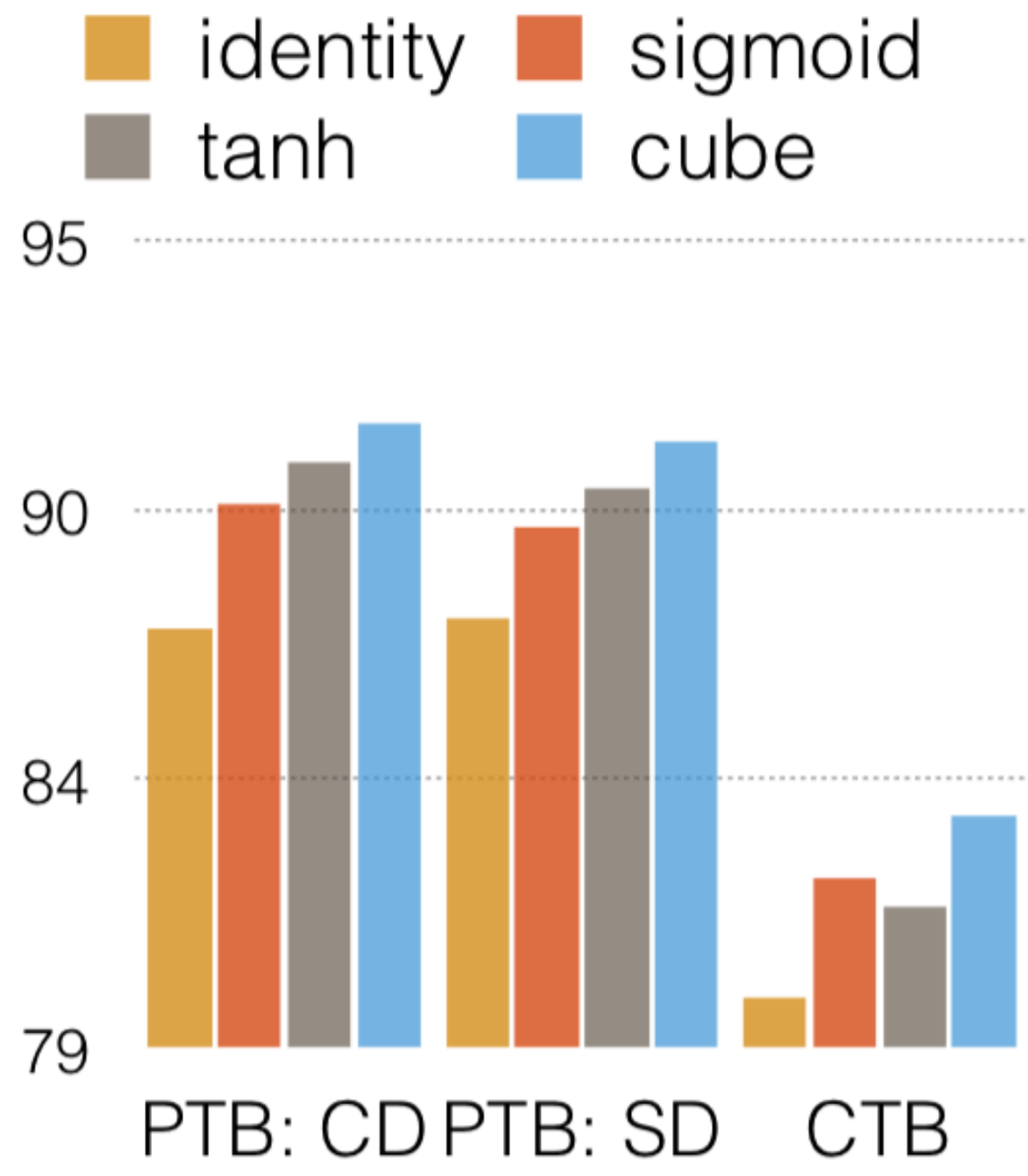


Parsing Speed (sent/s)

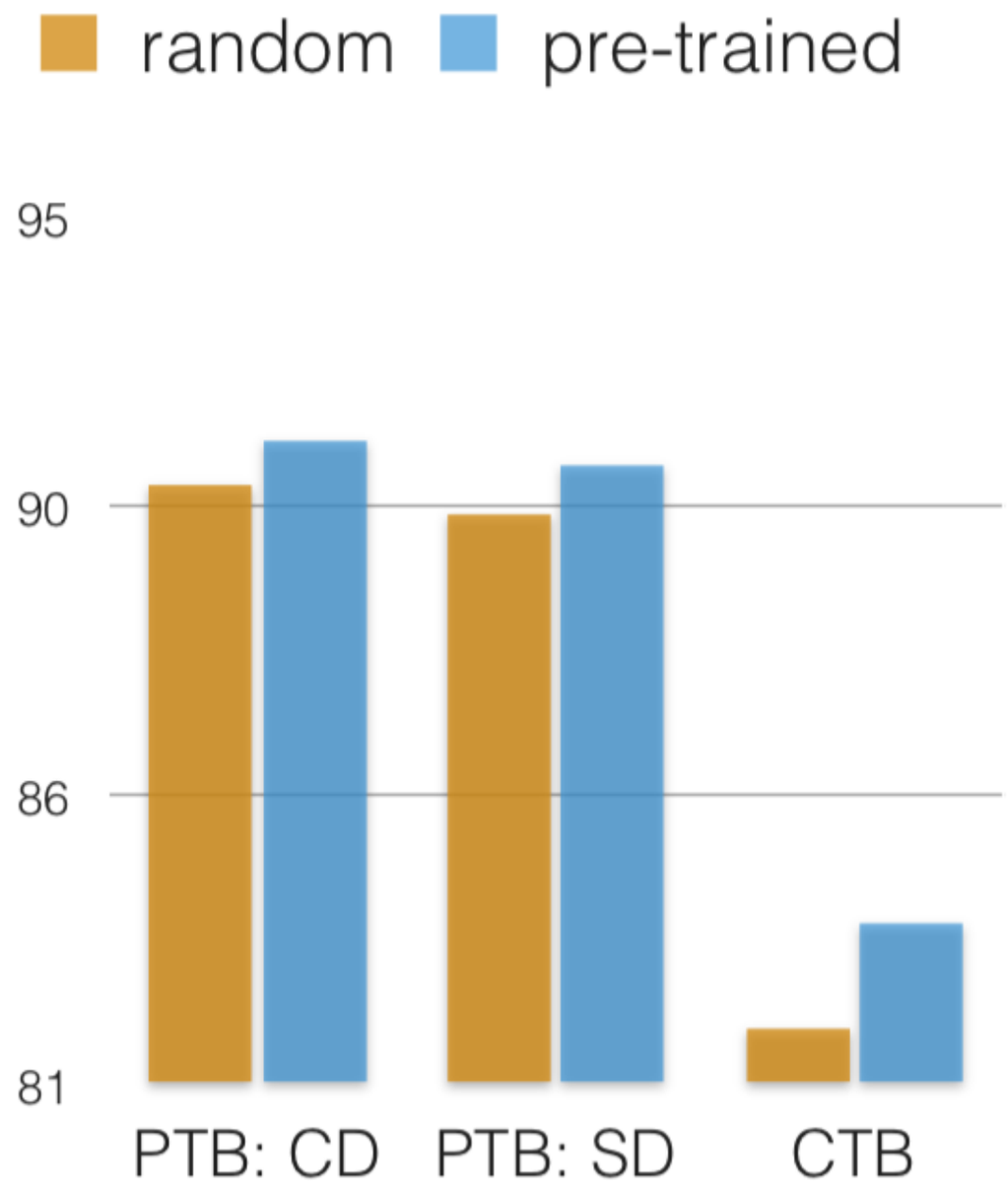
- Standard / eager
- Malt (stackproj / nirveeager)
- MST
- Our Parser



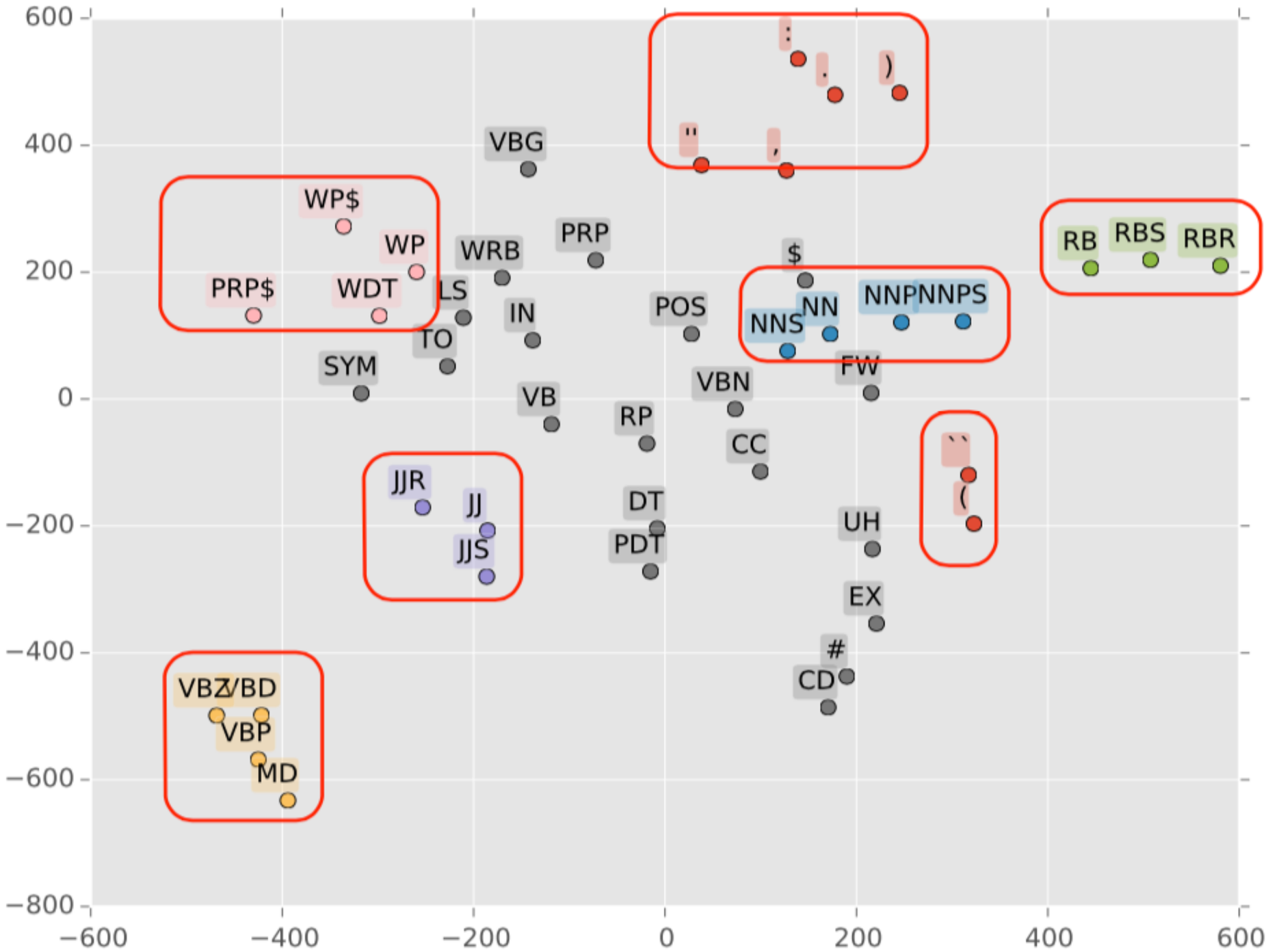
Cube Activation Function



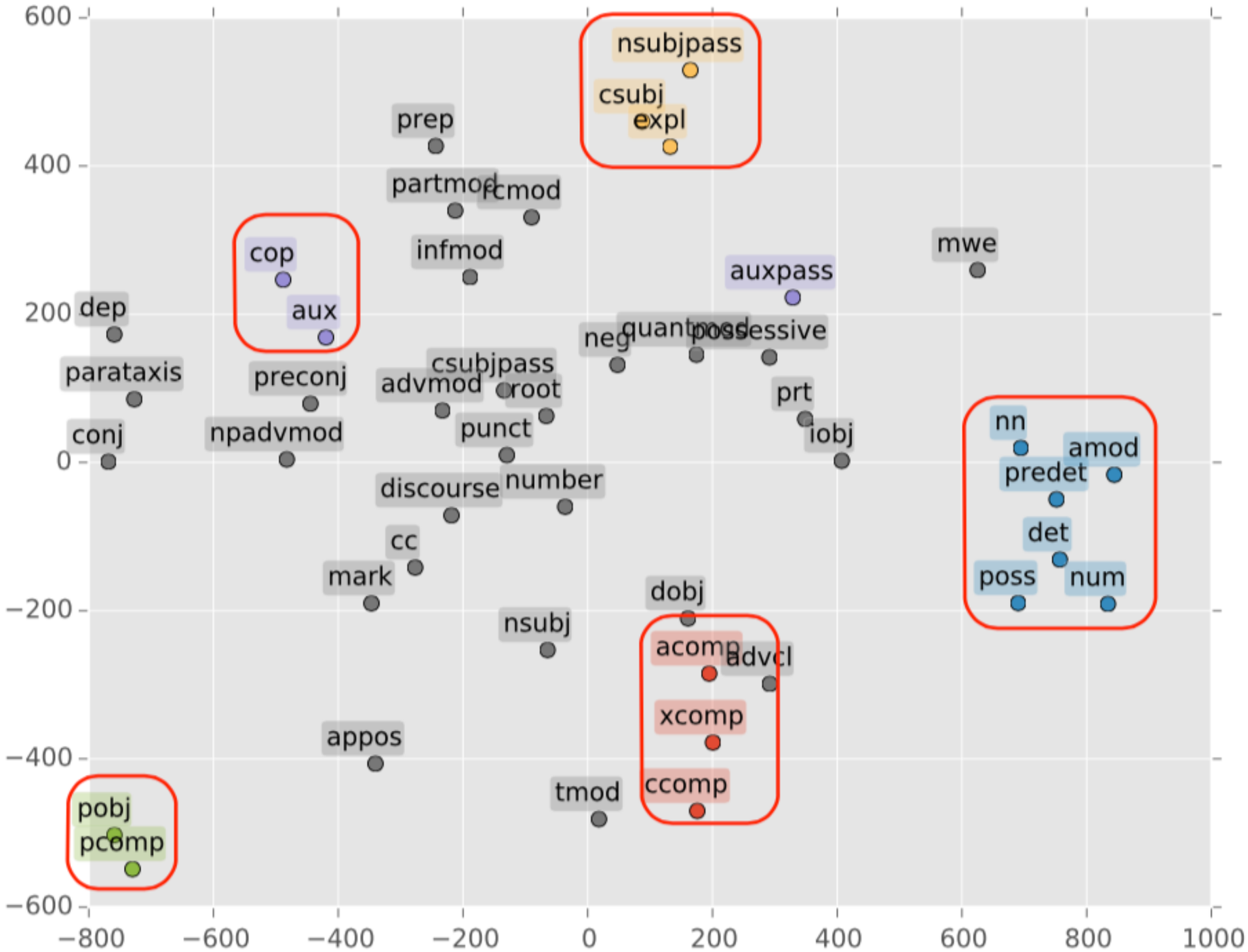
Pre-trained Word Vectors



POS Embeddings



Dependency Embeddings



Summary

- Transition-based parser using NNs
- State-of-the-art accuracy and speed
- Introduced POS / dep. embeddings, and cube activation function

Future Work

- Richer features (lemma, morph, distance, etc)
- Beam search
- Dynamic oracle