Natural Language Understanding

Lecture 9: Dependency Parsing with Neural Networks

Frank Keller

School of Informatics
University of Edinburgh
keller@inf.ed.ac.uk

February 13, 2017

Reading: Chen and Manning (2014).

Dependency Parsing

Traditional dependency parsing (Nivre 2003):

- simple shift-reduce parser (see last lecture);
- classifier chooses which transition (parser action) to take for each word in the input sentence;
- features for classifier similar to MALT parser (last lecture):
  - word/PoS unigrams, bigrams, trigrams;
  - state of the parser;
  - dependency tree built so far.

Problems:

- feature templates need to be handcrafted;
- results in millions of features
- feature are sparse and slow to extract.

Dependency Parsing

Chen and Manning (2014) propose:

- keep the simple shift-reduce parser;
- replace the classifier for transitions with a neural net;
- use dense features (embeddings) instead of sparse, handcrafted features.

Results:

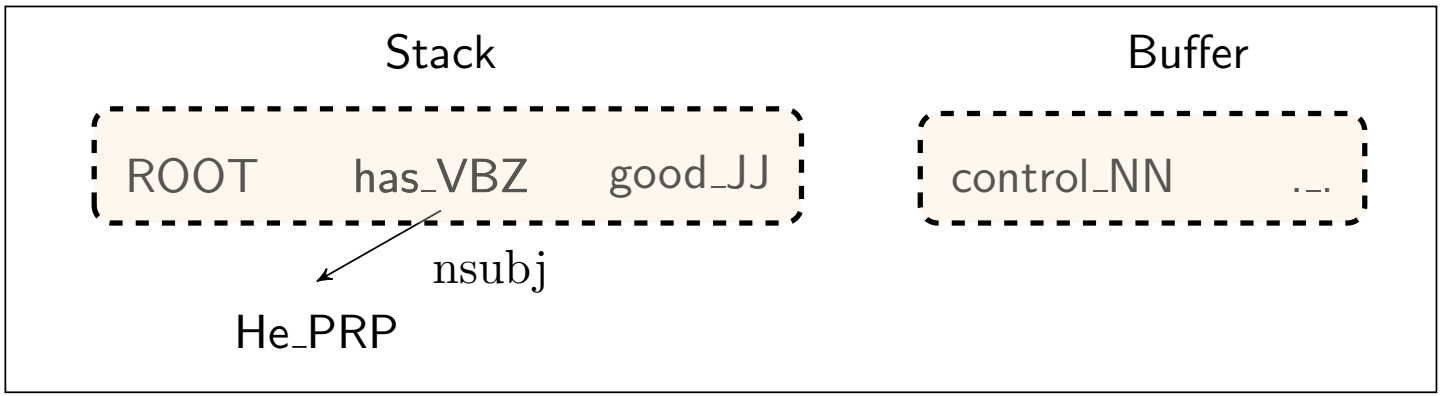
- efficient parser (up to twice as fast as standard MALT parser);
- good performance (about 2% higher precision than MALT).

Network Architecture

Goal of the network: predict correct transition $t \in \mathcal{T}$, based on configuration c. Relevant information:

1. words and PoS tags (e.g., has/VBZ);
2. head of words with dependency label (e.g., `nsubj`, `dobj`);
3. position of words on stack and buffer.

Correct transition: `SHIFT`



Network Architecture

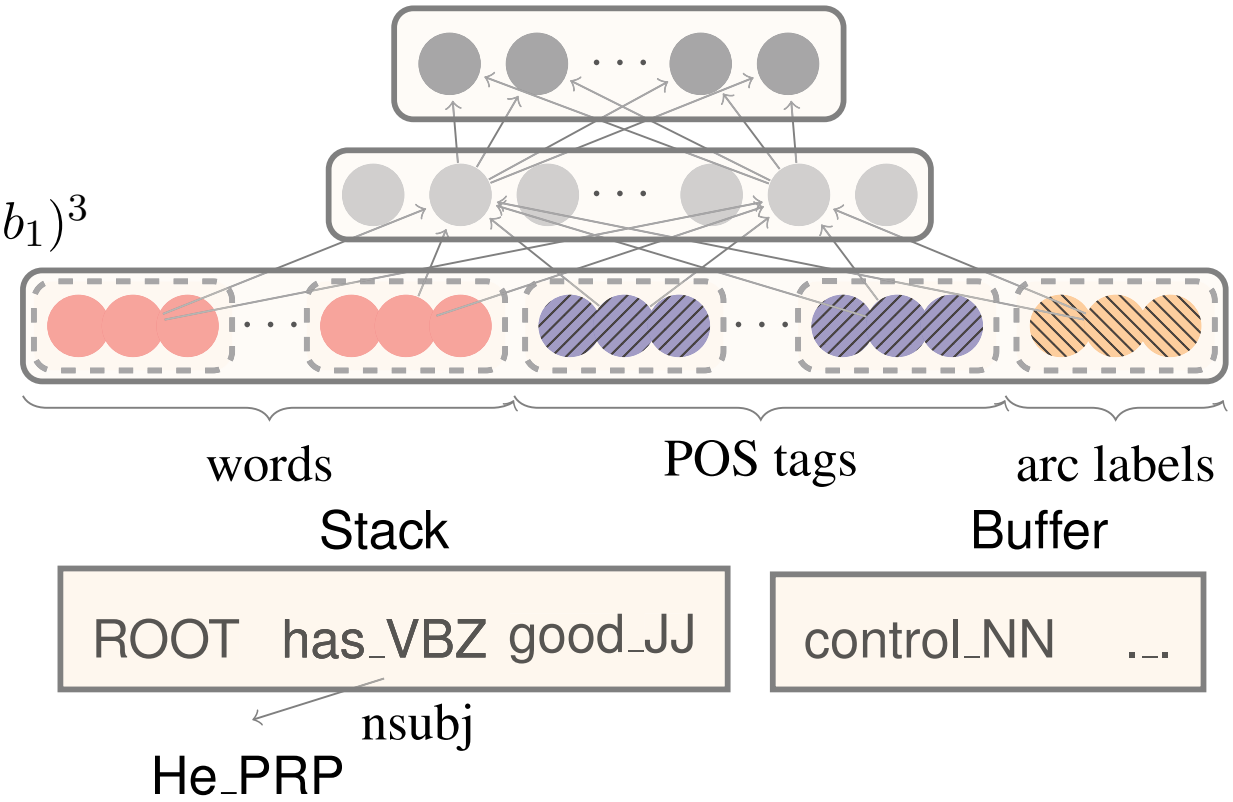
Softmax layer:

$$p = \texttt{softmax}(W_2 h)$$

Hidden layer:

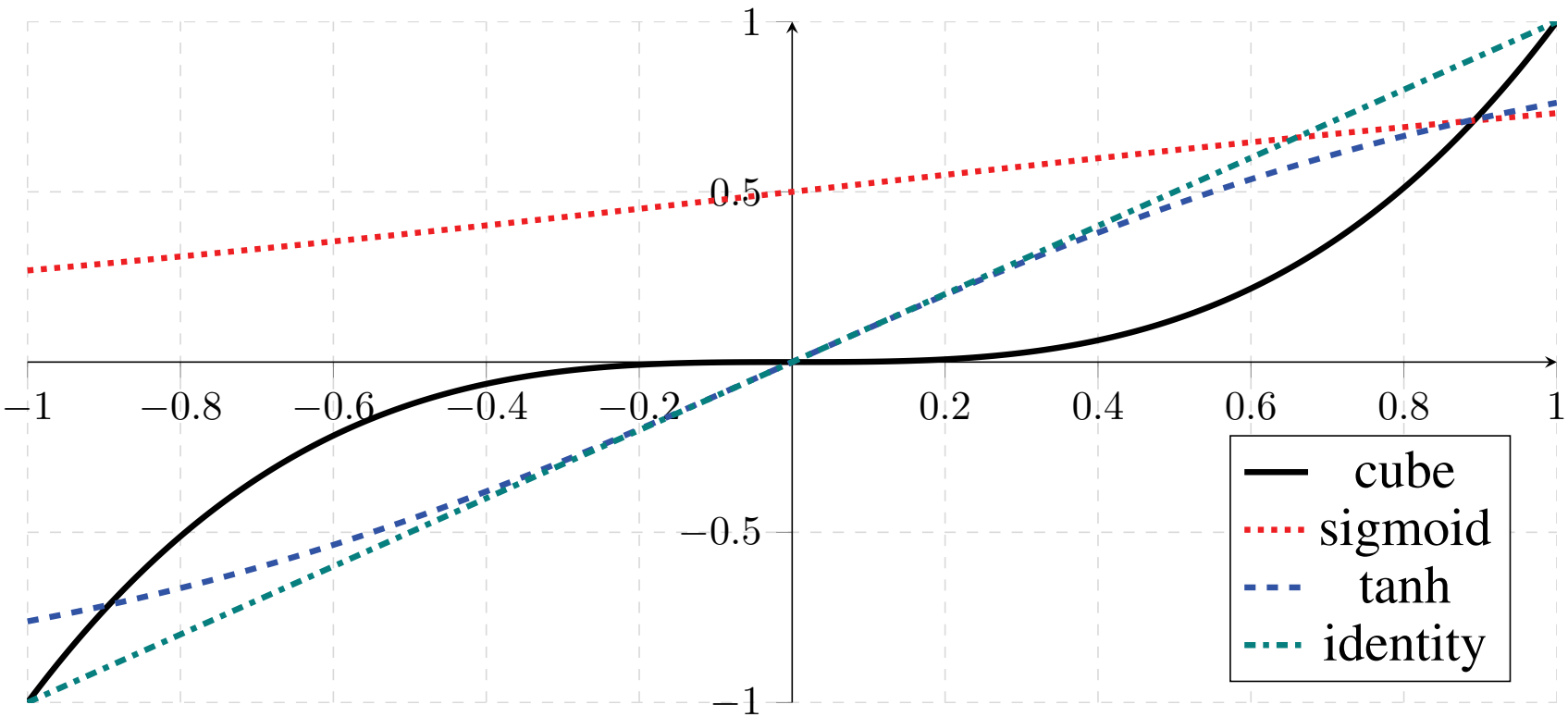
$$h = (W_1^w x^w + W_1^t x^t + W_1^l x^l + b_1)^3$$

Input layer: $[x^w, x^t, x^l]$

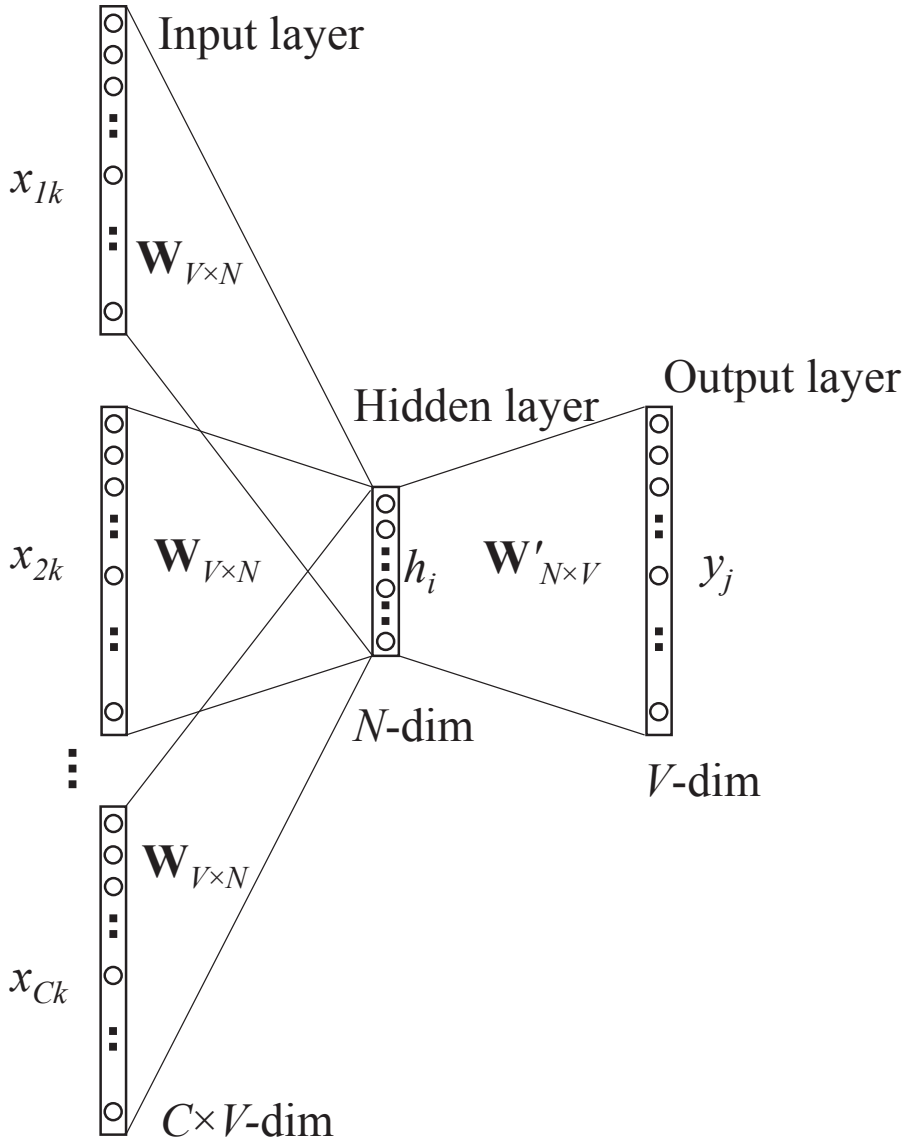


Configuration

Activation Function



Revision: Embeddings

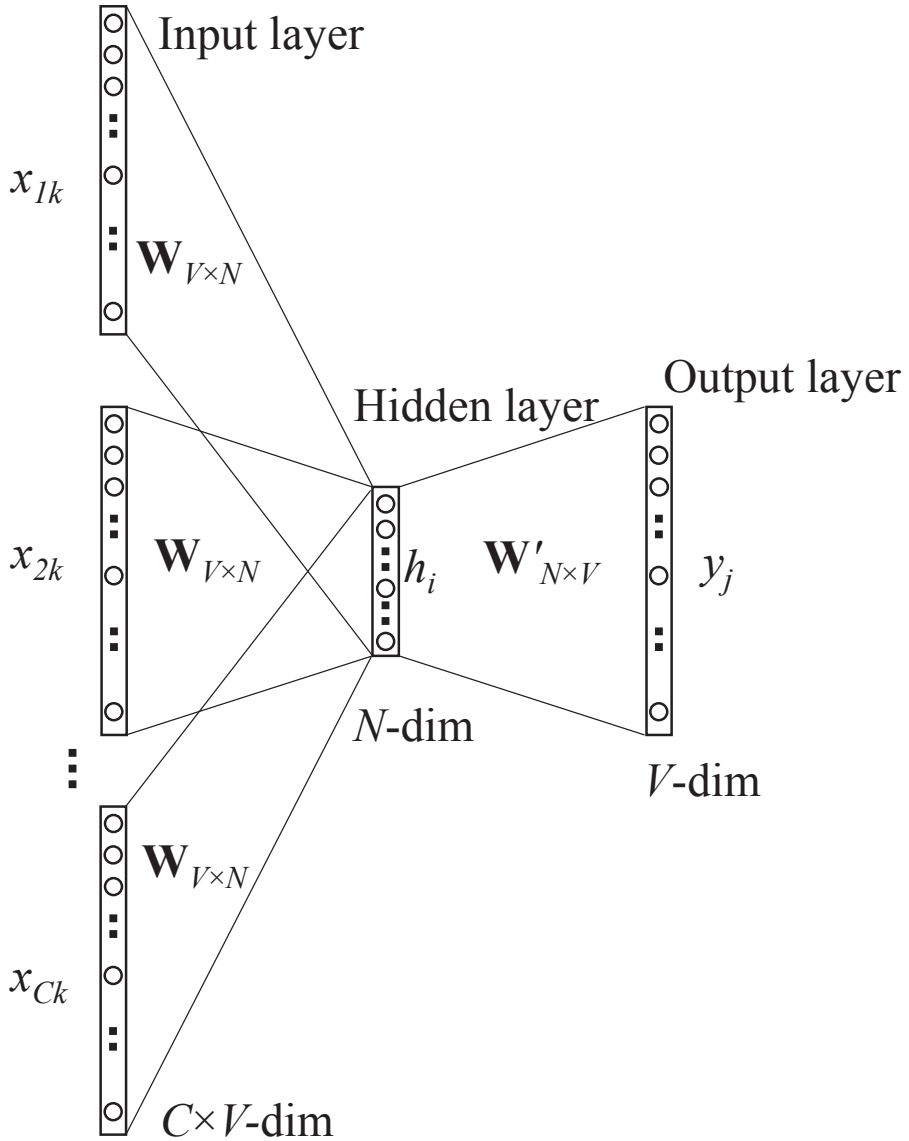


CBOW (Mikolov et al. 2013):

x_{ik}	context words (one-hot)
h_i	hidden units
y_j	output units (one-hot)
$\mathbf{W}$, $\mathbf{W}'$	weight matrices
V	vocabulary size
N	size of hidden layer
C	number of context words

[Figure from Rong (2014).]

Revision: Embeddings



CBOW (Mikolov et al. 2013):

x_{ik}	context words (one-hot)
h_i	hidden units
y_j	output units (one-hot)
$\mathbf{W}$, $\mathbf{W}'$	weight matrices
V	vocabulary size
N	size of hidden layer
C	number of context words

By *embedding* we mean the hidden layer **h**!

[Figure from Rong (2014).]

Embeddings

Chen and Manning (2014) use the following word embeddings S^w (18 elements):

1. top three words on stack and buffer: s_1, s_2, s_3, b_1, b_2, b_3;
2. first and second leftmost/rightmost children of top two words on stack: $lc_1(s_i)$, $rc_1(s_i)$, $lc_2(s_i)$, $rc_2(s_i)$, $i = 1, 2$;
3. leftmost of leftmost/rightmost of rightmost children of top two words on the stack: $lc_1(lc_1(s_i))$, $rc_1(rc_1(s_i))$, $i = 1, 2$.

Tag embeddings S^t (18 elements): same as word embeddings.

Arc label embeddings S^l (12 elements): same as word embeddings, excluding those the six words on the stack/buffer.

Training

Generate examples $\{(c_i, t_i)\}_{i=1}^{m}$ from sentences with gold parse trees using *shortest stack* oracle (always prefers LEFT-ARC(l) over SHIFT), where c_i is a configuration, $t_i \in \mathcal{T}$ a transition.

Objective: minimize cross-entropy loss with l_2-regularization:

$$L(\theta) = -\sum_i \log p_{t_i} + \frac{\lambda}{2}||\theta||^2$$

where p_{t_i} is the probability of transition t_i (from softmax layer), and θ is set of all parameters $\{W_1^w, W_1^t, W_1^l, b_1, W_2, E^w, E^t, E^l\}$.

Training

Use pre-trained word embeddings to initialize E^w; use random initialization within $(-0.01, 0.01)$ for E^t and E^l.

Word embeddings (Collobert et al. 2011) for English; 50-dimensional word2vec embeddings (Mikolov et al. 2013) for Chinese; compare with random initialization of E_w.

Mini-batched AdaGrad for optimization, dropout with 0.5 rate. Tune parameters on development set based UAS.

Hyper-parameters: embedding size $d = 50$, hidden layer size $h = 200$, regularization parameter $\lambda = 10^{-8}$, initial learning rate of AdaGrad $\alpha = 0.01$.

Decoding

The parser performs greedy decoding:

- for each parsing step, extract all word, PoS, and label embeddings from current configuration c;
- compute the hidden layer $h(c)$;
- pick transition with the highest score: $t = \text{argmax}_t\, W_2(t, \cdot)h(c)$;
- execute transition $c \rightarrow t(c)$.

Results: English with CoNLL Dependencies

Parser	Dev		Test		Speed
	UAS	LAS	UAS	LAS	(sent/s)
standard	89.9	88.7	89.7	88.3	51
eager	90.3	89.2	89.9	88.6	63
Malt:sp	90.0	88.8	89.9	88.5	560
Malt:eager	90.1	88.9	90.1	88.7	535
MSTParser	92.1	90.8	92.0	90.5	12
Our parser	92.2	91.0	92.0	90.7	1013

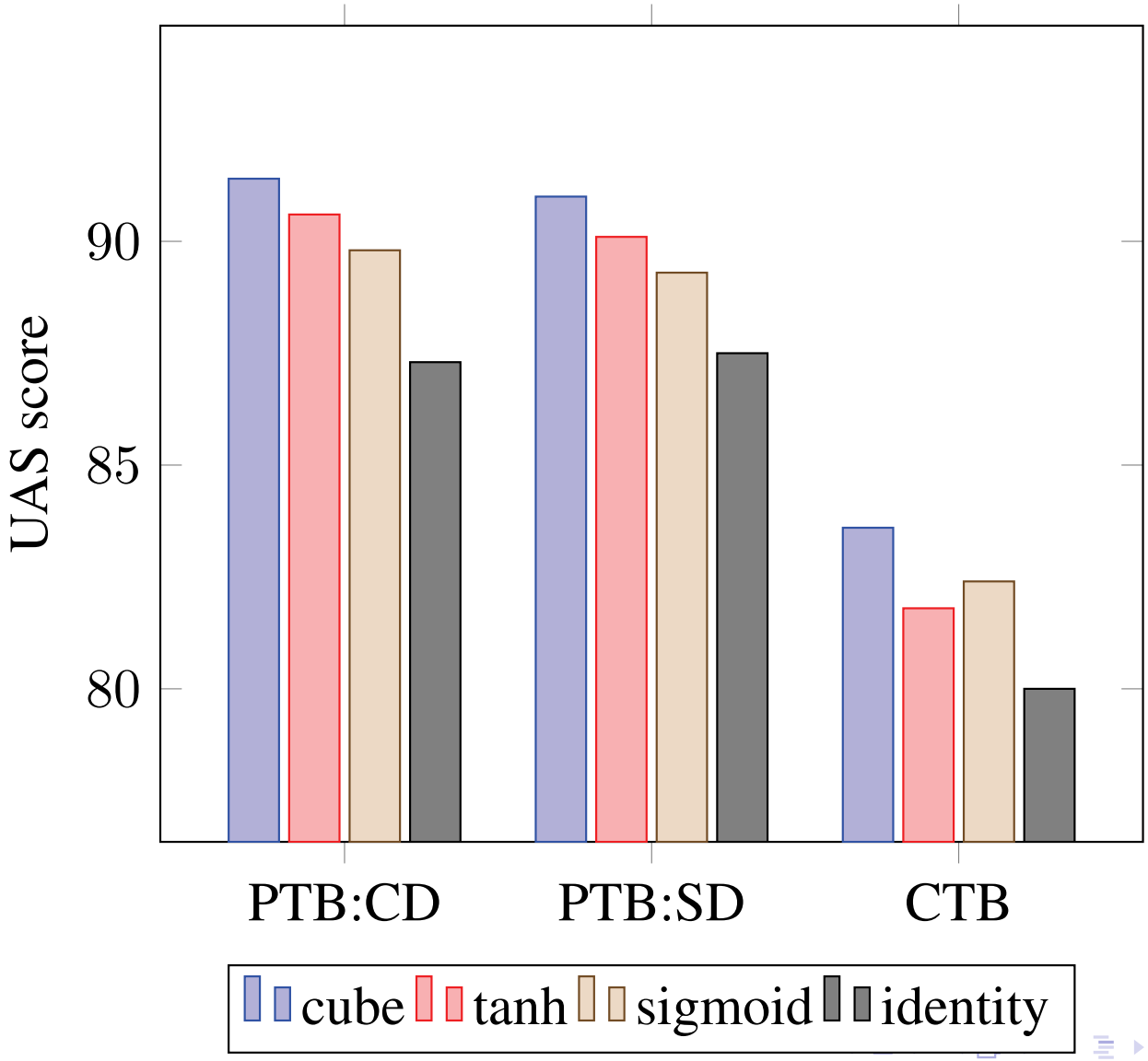
Results: English with Stanford Dependencies

Parser	Dev		Test		Speed
	UAS	LAS	UAS	LAS	(sent/s)
standard	90.2	87.8	89.4	87.3	26
eager	89.8	87.4	89.6	87.4	34
Malt:sp	89.8	87.2	89.3	86.9	469
Malt:eager	89.6	86.9	89.4	86.8	448
MSTParser	91.4	88.1	90.7	87.6	10
Our parser	92.0	89.7	91.8	89.6	654

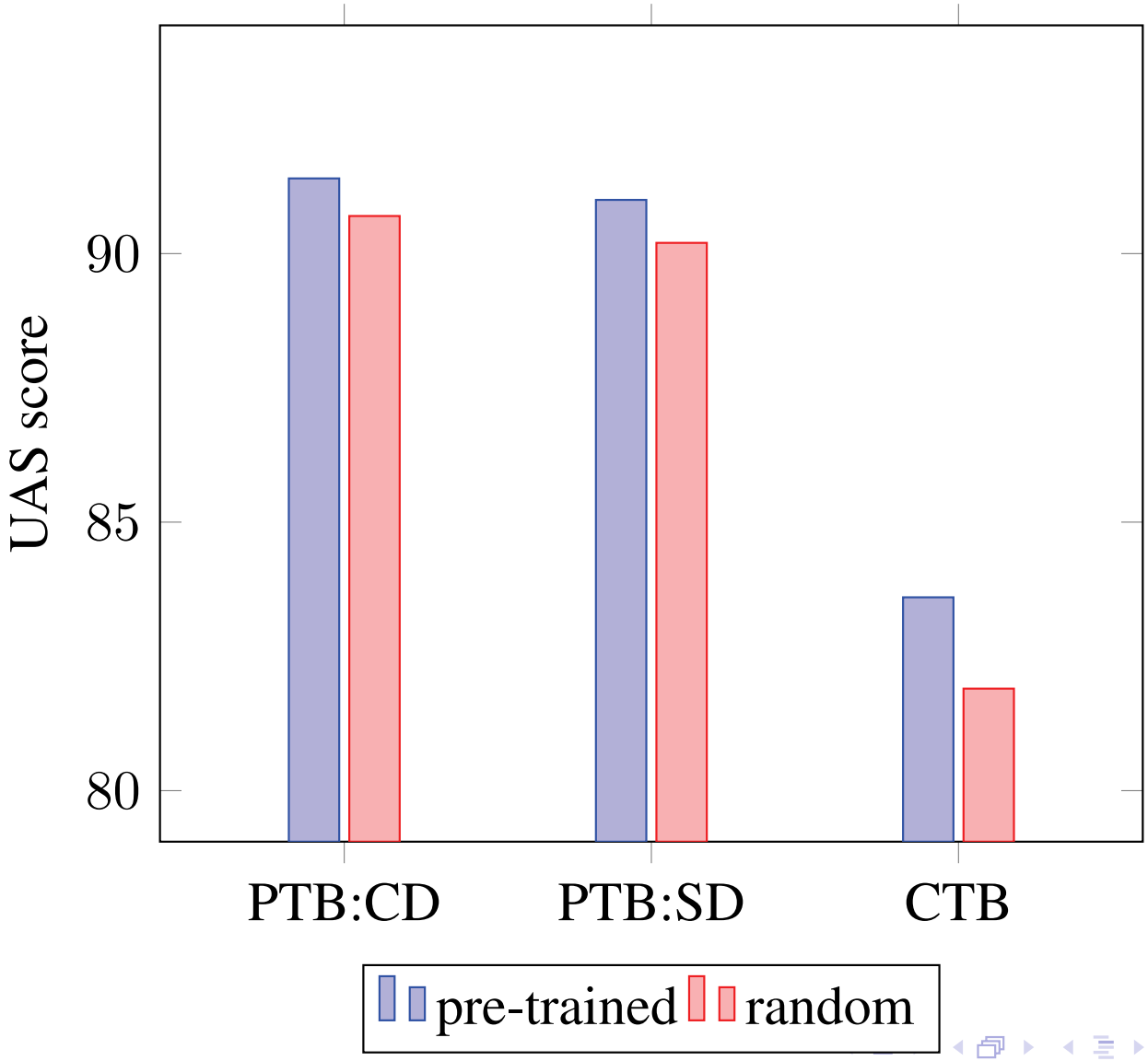
Results: Chinese

Parser	Dev		Test		Speed
	UAS	LAS	UAS	LAS	(sent/s)
standard	82.4	80.9	82.7	81.2	72
eager	81.1	79.7	80.3	78.7	80
Malt:sp	82.4	80.5	82.4	80.6	420
Malt:eager	81.2	79.3	80.2	78.4	393
MSTParser	84.0	82.1	83.0	81.2	6
Our parser	84.0	82.4	83.9	82.4	936

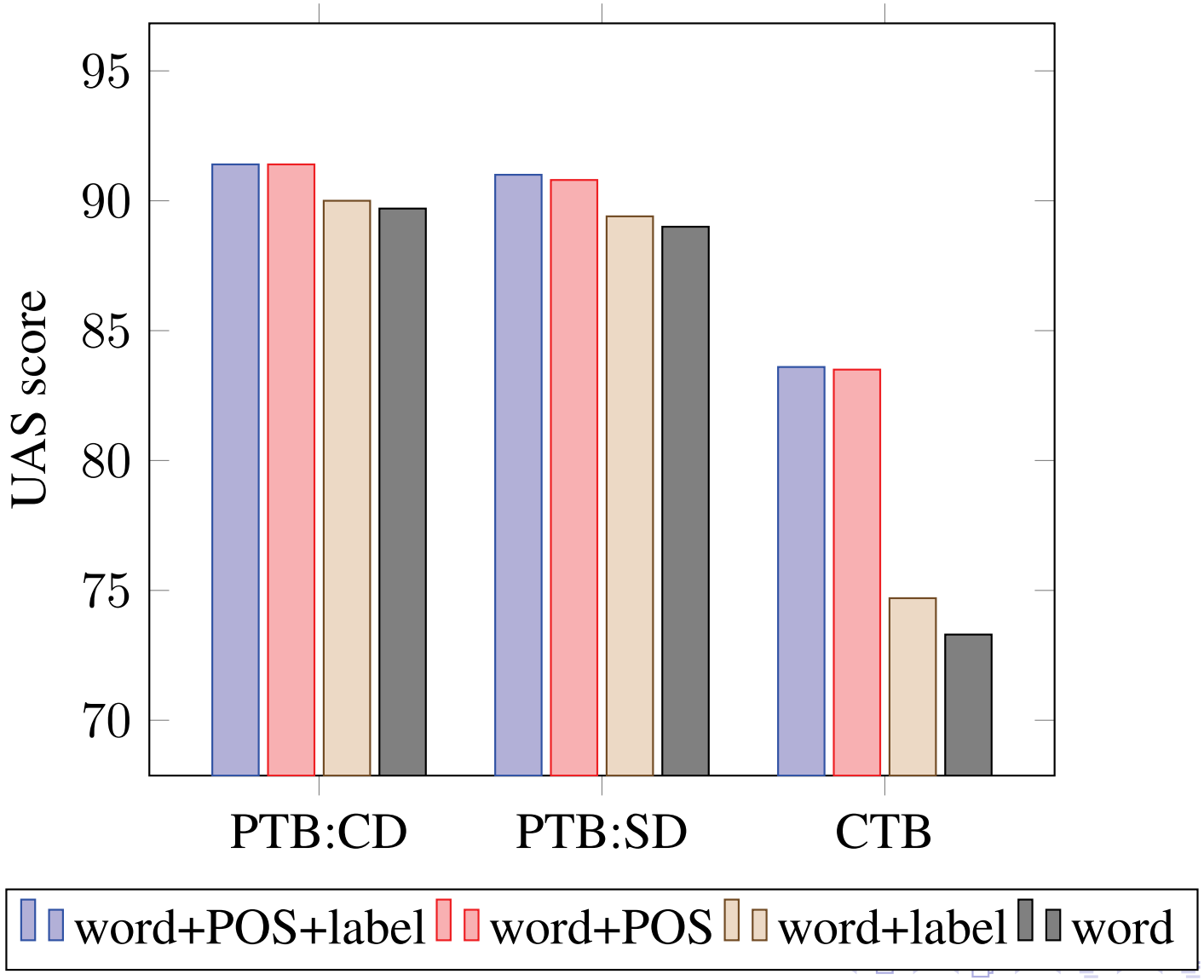
Effect of Activation Function



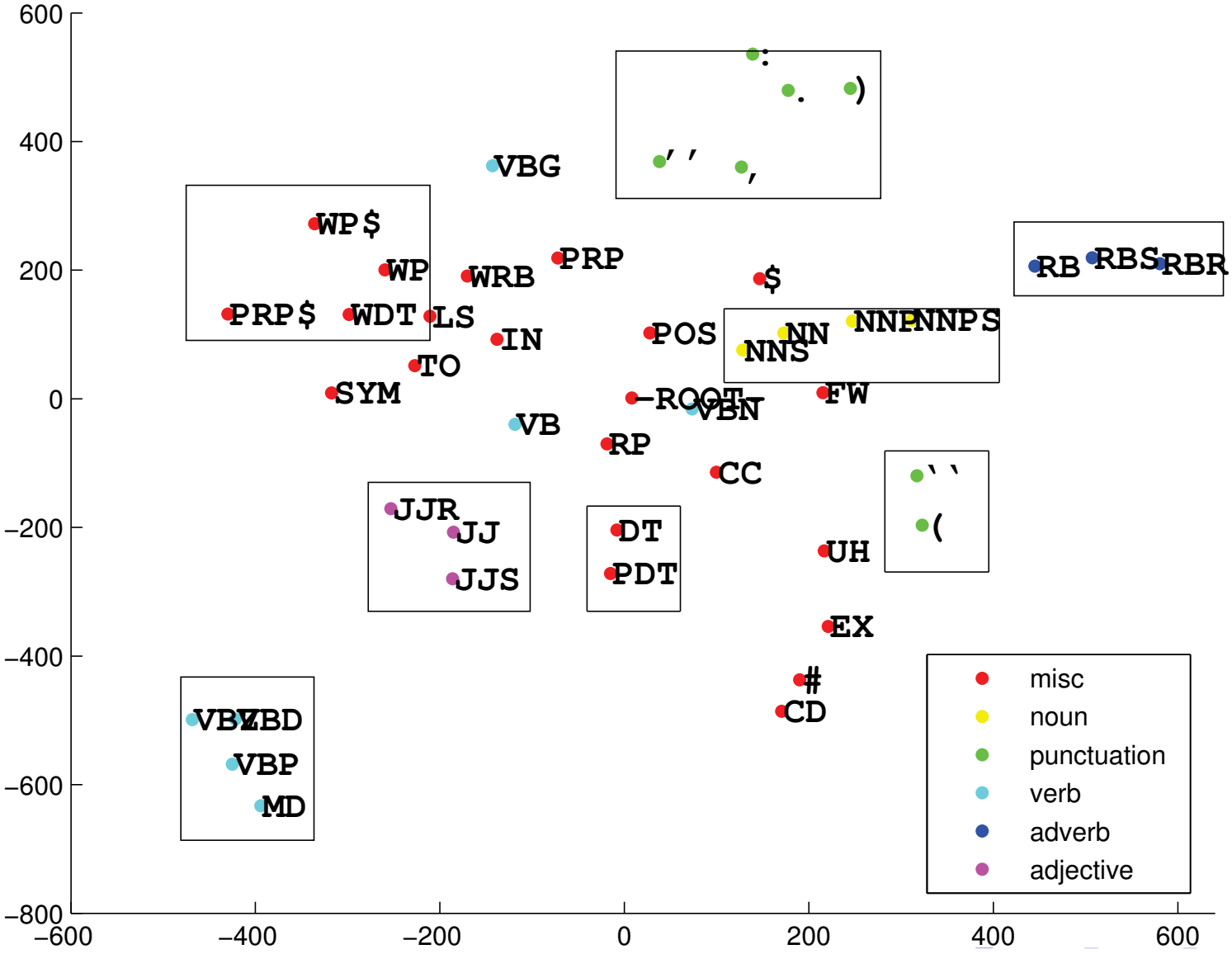
Pre-trained Embeddings vs. Random Initialization



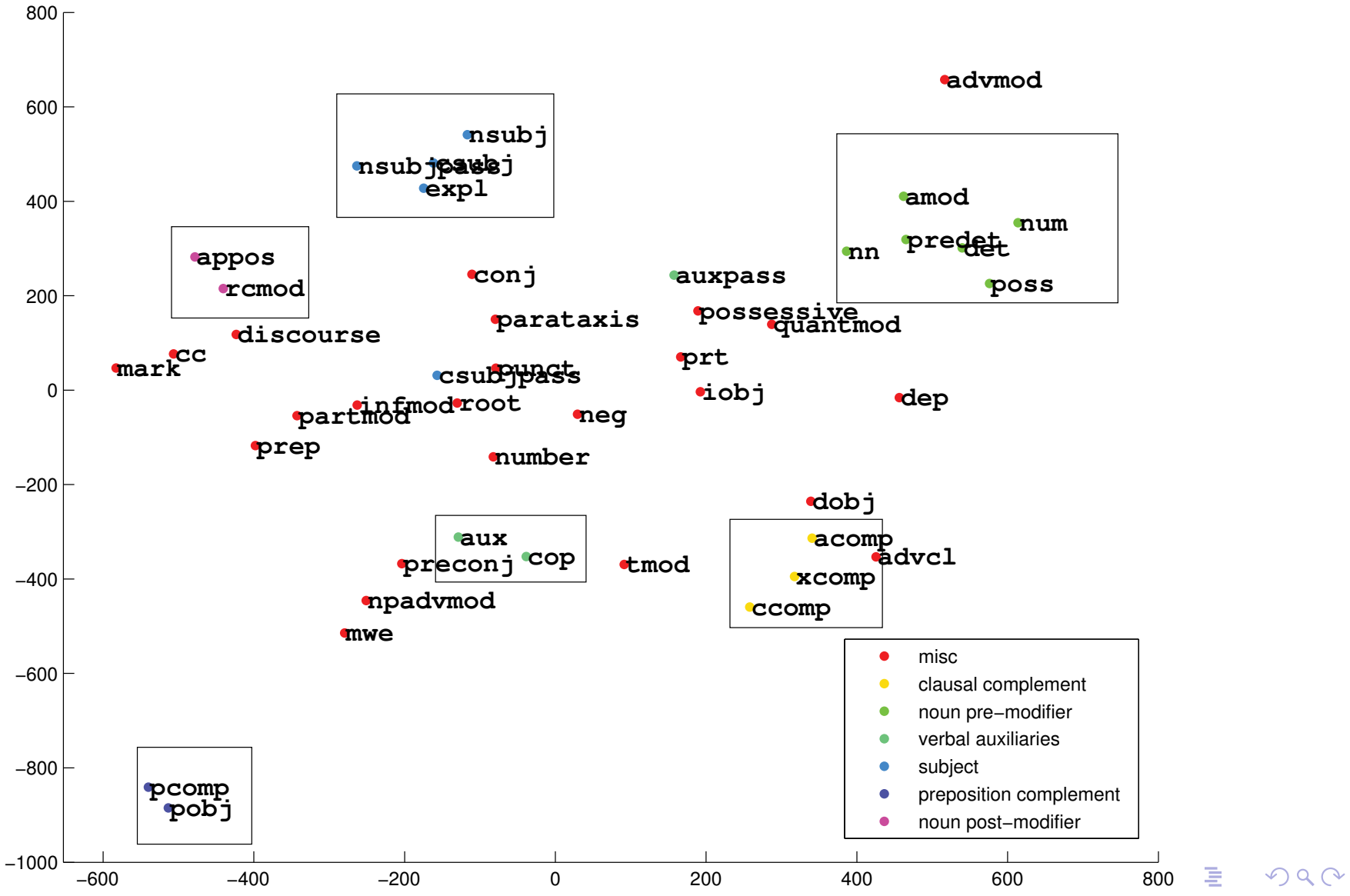
Effect of PoS and Label Embeddings



Visualization of PoS Embeddings



Visualization Label Embeddings



Summary

- Chen and Manning's (2014) model builds on standard transition-based dependency parsing;
- uses neural net to select transitions;
- uses dense features (embeddings) instead of sparse, handcrafted features;
- embeddings over words, PoS, and arc labels;
- new cube activation function;
- good accuracy for English and Chinese dependency parsing;
- substantial improvement in speed.

References

Chen, Danqi, and Christopher D. Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 740–750. Doha.

Collobert, Ronan, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 12: 2493–2537.

Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, eds., *Advances in Neural Information Processing Systems 26*, 3111–3119. Red Hook, NY: Curran Associates.

Nivre, Joakim. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of the International Workshop on Parsing Technologies*, 149–160. Nancy.

Rong, Xin. 2014. word2vec parameter learning explained. Unpublished mansucript, arXiv:1411.2738.