

# Natural Language Processing with Deep Learning

## CS224N/Ling284



**Christopher Manning** and Richard Socher  
Lecture 6: Dependency Parsing



# Lecture Plan

1. Syntactic Structure: Consistency and Dependency
2. Dependency Grammar
3. *Research highlight*
4. Transition-based dependency parsing
5. Neural dependency parsing

## Reminders/comments:

Assignment 1 due tonight 😊

Assignment 2 out today 😞

Final project discussion – come meet with us

Sorry that we've been kind of overloaded for office hours



# 1. Two views of linguistic structure:

## Constituency = phrase structure grammar = context-free grammars (CFGs)

Phrase structure organizes words into nested constituents.

the	cat
a	dog
large	in a crate
barking	on the table
cuddly	by the door



# 1. Two views of linguistic structure:

## Constituency = phrase structure grammar = context-free grammars (CFGs)

Phrase structure organizes words into nested constituents.

the cat

a dog

large

in a crate

barking

on the table

cuddly

by the door

large barking

talk to

look for

$NP \rightarrow Det\ N$

$NP \rightarrow Det\ (A)\ N\ (PP)$

$PP \rightarrow P\ NP$

$NP \rightarrow Det\ A^*\ N\ PP^*$

$VP \rightarrow V\ PP$



# Two views of linguistic structure: Dependency structure

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

*Look for the large barking dog by the door in a crate*



# Two views of linguistic structure:

## Dependency structure

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





# Ambiguity: PP attachments

Scientists study whales from space



## Ambiguity: PP attachments

Scientists study whales from space

A diagram illustrating the ambiguity of prepositional phrase (PP) attachment for the sentence "Scientists study whales from space". The words are arranged horizontally. Three curved arrows originate from the words: a black arrow from "study" points to "Scientists"; a black arrow from "whales" points to "study"; and a black arrow from "space" points to "whales". A long, thick red curved arrow originates from "study" and points to "space", representing the attachment of the PP "from space" to the verb "study".

Scientists study whales from space

A second diagram illustrating the ambiguity of PP attachment for the same sentence. It uses the same horizontal arrangement of words and the same three black arrows as the first diagram. However, the thick red curved arrow originates from "from" and points to "space", representing the attachment of the PP "from space" to the noun "whales".





# Attachment ambiguities

- A key parsing decision is how we ‘attach’ various constituents
  - PPs, adverbial or participial phrases, infinitives, coordinations,

The board approved [its acquisition] [by Royal Trustco Ltd.]  
[of Toronto]  
[for \$27 a share]  
[at its monthly meeting].



# Attachment ambiguities

- A key parsing decision is how we 'attach' various constituents
  - PPs, adverbial or participial phrases, infinitives, coordinations,

The board approved [its acquisition] [by Royal Trustco Ltd.]  
[of Toronto] [for \$27 a share]  
[at its monthly meeting].

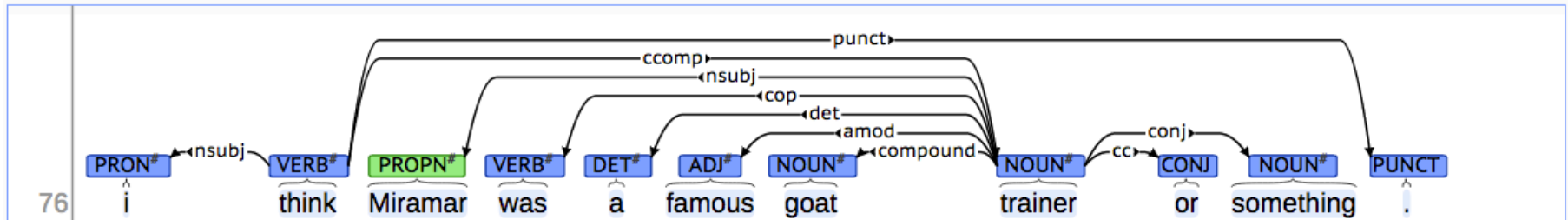
- Catalan numbers:  $C_n = (2n)! / [(n+1)!n!]$
- An exponentially growing series, which arises in many tree-like contexts:
  - E.g., the number of possible triangulations of a polygon with  $n+2$  sides
    - Turns up in triangulation of probabilistic graphical models....



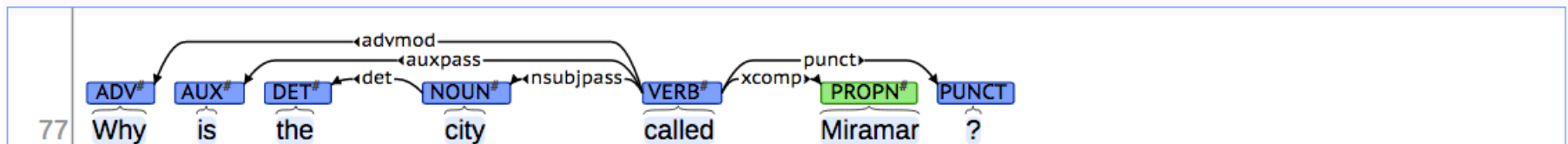
# The rise of annotated data: Universal Dependencies treebanks

[Universal Dependencies: <http://universaldependencies.org/> ;  
cf. Marcus et al. 1993, The Penn Treebank, *Computational Linguistics*]

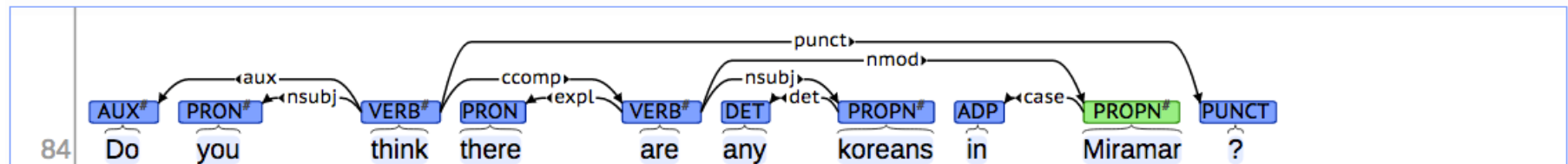
[context] [conllu]



[context] [conllu]



[context] [conllu]





# The rise of annotated data

Starting off, building a treebank seems a lot slower and less useful than building a grammar

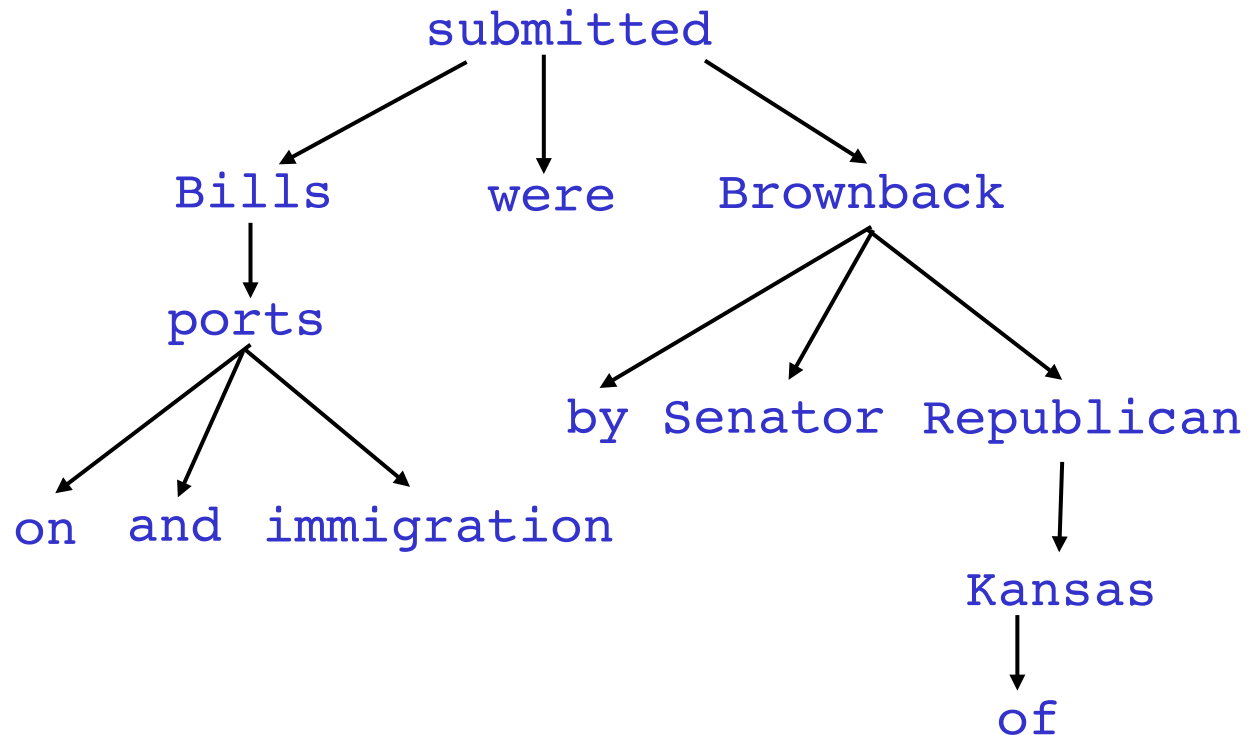
But a treebank gives us many things

- Reusability of the labor
  - Many parsers, part-of-speech taggers, etc. can be built on it
  - Valuable resource for linguistics
- Broad coverage, not just a few intuitions
- Frequencies and distributional information
- A way to evaluate systems



## 2. Dependency Grammar and Dependency Structure

Dependency syntax postulates that syntactic structure consists of relations between lexical items, normally binary asymmetric relations (“arrows”) called **dependencies**

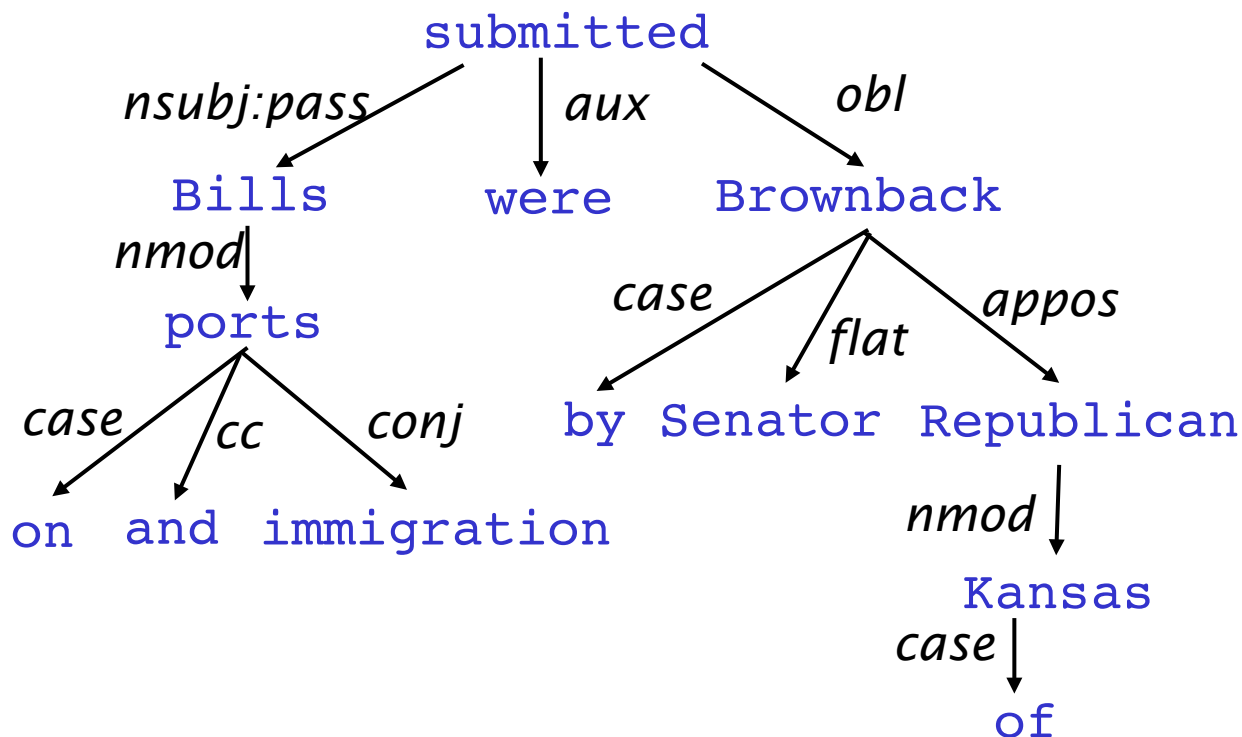




# Dependency Grammar and Dependency Structure

Dependency syntax postulates that syntactic structure consists of relations between lexical items, normally binary asymmetric relations (“arrows”) called **dependencies**

The arrows are commonly **typed** with the name of grammatical relations (subject, prepositional object, apposition, etc.)



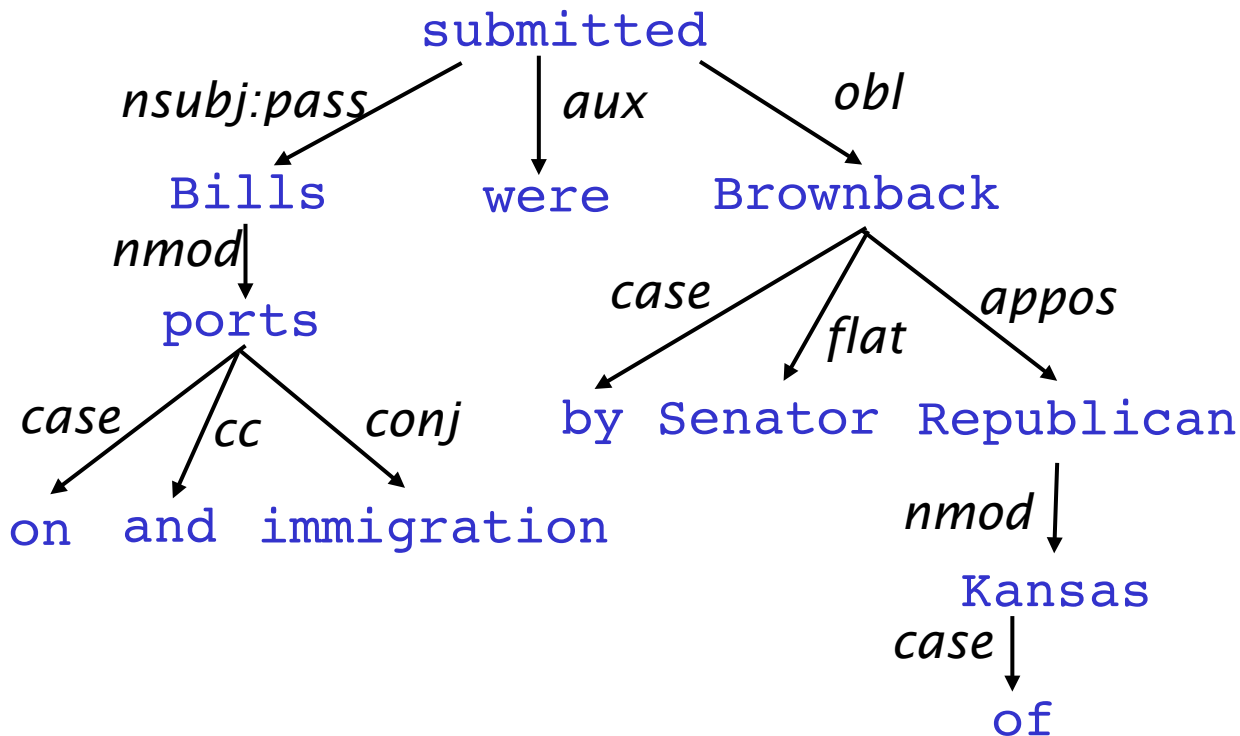


# Dependency Grammar and Dependency Structure

Dependency syntax postulates that syntactic structure consists of relations between lexical items, normally binary asymmetric relations (“arrows”) called **dependencies**

The arrow connects a **head** (governor, superior, regent) with a **dependent** (modifier, inferior, subordinate)

Usually, dependencies form a tree (connected, acyclic, single-head)







# Pāṇini's grammar (c. 5th century BCE)



Gallery: <http://wellcomeimages.org/indexplus/image/L0032691.html>  
CC BY 4.0 File: Birch bark MS from Kashmir of the Rupavatra Wellcome L0032691.jpg



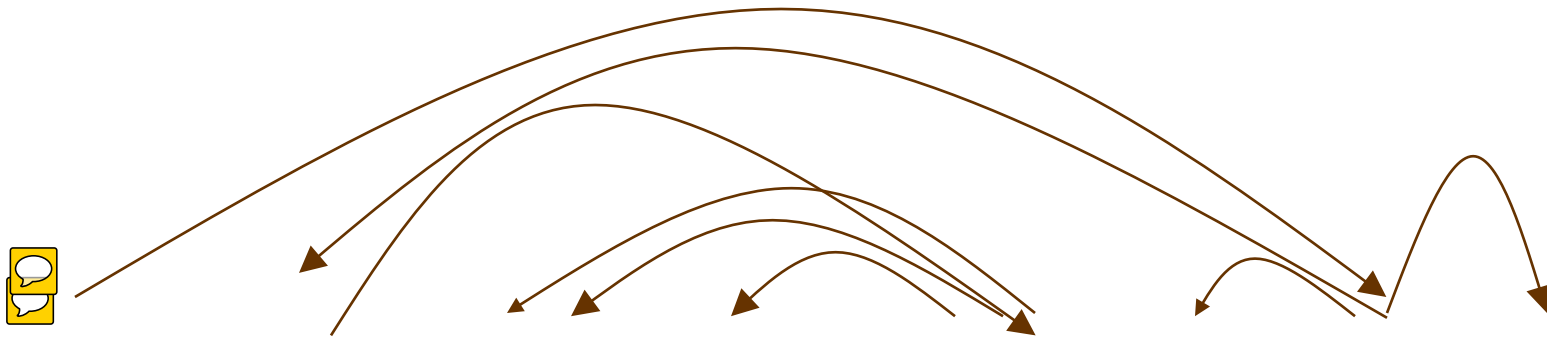


# Dependency Grammar/Parsing History

- The idea of dependency structure goes back a long way
  - To Pāṇini's grammar (c. 5th century BCE)
  - Basic approach of 1st millennium Arabic grammarians
- Constituency/context-free grammars is a new-fangled invention
  - 20th century invention (R.S. Wells, 1947)
- Modern dependency work often linked to work of L. Tesnière (1959)
  - Was dominant approach in "East" (Russia, China, ...)
    - Good for free-er word order languages
- Among the earliest kinds of parsers in NLP, even in the US:
  - David Hays, one of the founders of U.S. computational linguistics, built early (first?) dependency parser (Hays 1962)



# Dependency Grammar and Dependency Structure



ROOT Discussion of the outstanding issues was completed .



- Some people draw the arrows one way; some the other way!
  - Tesnière had them point from head to dependent...
- Usually add a fake ROOT so every word is a dependent of precisely 1 other node



# Dependency Conditioning Preferences



What are the sources of information for dependency parsing?

1. Bilexical affinities [discussion → issues] is plausible 
2. Dependency distance mostly with nearby words
3. Intervening material 

Dependencies rarely span intervening verbs or punctuation


4. Valency of heads

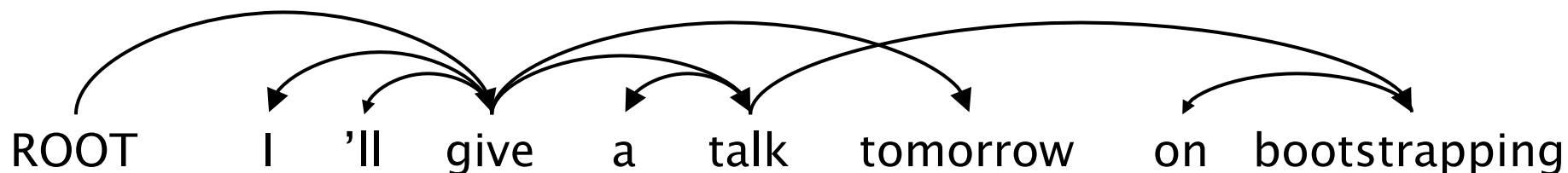
How many dependents on which side are usual for a head?





# Dependency Parsing

- A sentence is parsed by choosing for each word what other word (including ROOT) is it a dependent of
- Usually some constraints:
  - Only one word is a dependent of ROOT
  - Don't want cycles  $A \rightarrow B, B \rightarrow A$
- This makes the dependencies a tree
- Final issue is whether arrows can cross (**non-projective**) or not 





# Methods of Dependency Parsing

## 1. Dynamic programming

Eisner (1996) gives a clever algorithm with complexity  $O(n^3)$ , by producing parse items with heads at the ends rather than in the middle

## 2. Graph algorithms

You create a Minimum Spanning Tree for a sentence

McDonald et al.'s (2005) MSTParser scores dependencies independently using an ML classifier (he uses MIRA, for online learning, but it can be something else)

## 3. Constraint Satisfaction

Edges are eliminated that don't satisfy hard constraints. Karlsson (1990), etc.

## 4. "Transition-based parsing" or "deterministic dependency parsing"

Greedy choice of attachments guided by good machine learning classifiers

MaltParser (Nivre et al. 2008). Has proven highly effective.

# Improving Distributional Similarity with Lessons Learned from Word Embeddings

---

*Omer Levy, Yoav Goldberg, Ido Dagan*

Presented by: **Ajay Sohmshtetty**

# Word Representation Methods

**Count-based distributional models**

**Neural network-based models**

# Word Representation Methods

## Count-based distributional models

- SVD (Singular Value Decomposition)
- PPMI (Positive Pointwise Mutual Information)

## Neural network-based models

- SGNS (Skip-Gram Negative Sampling) / CBOW
- GloVe



# Word Representation Methods

## Count-based distributional models

- SVD (Singular Value Decomposition)
- PPMI (Positive Pointwise Mutual Information)

## Neural network-based models

- SGNS (Skip-Gram Negative Sampling) / CBOW
- GloVe

## Conventional wisdom:

Neural-network based models > Count-based models

# Word Representation Methods

## Count-based distributional models

- SVD (Singular Value Decomposition)
- PPMI (Positive Pointwise Mutual Information)

## Neural network-based models

- SGNS (Skip-Gram Negative Sampling) / CBOW
- GloVe

## Levy et. al.:

Hyperparameters and system design choices more important, not the embedding algorithms themselves.

# Hyperparameters in Skip-Gram

$$J_t(\theta) = \log \sigma(u_o^T v_c) + \sum_{i=1}^k \mathbb{E}_{j \sim P(w)} [\log \sigma(-u_j^T v_c)]$$

# of negative samples

$$P(w) = U(w)^{3/4} / Z$$

Unigram distribution smoothing exponent

→ These can be transferred over to the count-based variants.

# Context Distribution Smoothing

$$PMI_{\alpha}(w, c) = \log \frac{\hat{P}(w, c)}{\hat{P}(w) \hat{P}_{\alpha}(c)}$$

$$\hat{P}_{\alpha}(c) = \frac{\#(c)^{\alpha}}{\sum_c \#(c)^{\alpha}}$$

## Shifted PMI

$$SPPMI(w, c) = \max (PMI (w, c) - \log k, 0)$$

# All Transferable Hyperparameters

	Hyperparameter	Explored Values	Applicable Methods
<b>Preprocessing</b>	Window	2, 5, 10	All
	Dynamic Context Window	None, with	All
	Subsampling	None, dirty, clean	All
	Deleting Rare Words	None, with	All
<b>Association Metric</b>	Shifted PMI	1, 5, 15	PPMI, SVD, SGNS
	Context Distribution Smoothing	1, 0.75	PPMI, SVD, SGNS
	Adding Context Vectors	Only w, w+c	SVD, SGNS, GloVe
<b>Postprocessing</b>	Eigenvalue Weighting	0, 0.5, 1	SVD
	Vector Normalization	None, row, col, both	All

# Results

## Word Similarity Tasks

## Analogy Tasks

win	Method	Word Similarity Tasks						Analogy Tasks	
		WordSim Similarity	WordSim Relatedness	Bruni et al. MEN	Radinsky et al. M. Turk	Luong et al. Rare Words	Hill et al. SimLex	Google Add / Mul	MSR Add / Mul
2	PPMI	.732	<b>.699</b>	.744	.654	.457	.382	.552 / .677	.306 / .535
	SVD	.772	.671	<b>.777</b>	.647	<b>.508</b>	.425	.554 / .591	.408 / .468
	SGNS	<b>.789</b>	.675	.773	<b>.661</b>	.449	<b>.433</b>	.676 / <b>.689</b>	.617 / <b>.644</b>
	GloVe	.720	.605	.728	.606	.389	.388	.649 / .666	.540 / .591
5	PPMI	.732	<b>.706</b>	.738	<b>.668</b>	.442	.360	.518 / .649	.277 / .467
	SVD	.764	.679	<b>.776</b>	.639	<b>.499</b>	<b>.416</b>	.532 / .569	.369 / .424
	SGNS	<b>.772</b>	.690	.772	.663	.454	.403	.692 / <b>.714</b>	.605 / <b>.645</b>
	GloVe	.745	.617	.746	.631	.416	.389	.700 / .712	.541 / .599
10	PPMI	.735	<b>.701</b>	.741	.663	.235	.336	.532 / .605	.249 / .353
	SVD	.766	.681	.770	.628	<b>.312</b>	.419	.526 / .562	.356 / .406
	SGNS	<b>.794</b>	.700	<b>.775</b>	<b>.678</b>	.281	<b>.422</b>	.694 / .710	.520 / <b>.557</b>
	GloVe	.746	.643	.754	.616	.266	.375	.702 / <b>.712</b>	.463 / .519

# Key Takeaways

- This paper challenges the conventional wisdom that neural network-based models are superior to count-based models.
- While model design is important, hyperparameters are also KEY for achieving reasonable results. Don't discount their importance!
- Challenge the status quo!





## 4. Greedy transition-based parsing

[Nivre 2003]



- A simple form of greedy discriminative dependency parser
- The parser does a sequence of bottom up actions
  - Roughly like “shift” or “reduce” in a shift-reduce parser, but the “reduce” actions are specialized to create dependencies with head on left or right
- The parser has:
  - a stack  $\sigma$ , written with top to the right
    - which starts with the ROOT symbol
  - a buffer  $\beta$ , written with top to the left
    - which starts with the input sentence
  - a set of dependency arcs  $A$ 
    - which starts off empty
  - a set of actions



# Basic transition-based dependency parser

**Start:**  $\sigma = [\text{ROOT}]$ ,  $\beta = w_1, \dots, w_n$ ,  $A = \emptyset$

1. Shift  $\sigma, w_i | \beta, A \Rightarrow \sigma | w_i, \beta, A$

2. Left-Arc<sub>r</sub>  $\sigma | w_i | w_j, \beta, A \Rightarrow \sigma | w_j, \beta, A \cup \{r(w_j, w_i)\}$

3. Right-Arc<sub>r</sub>  $\sigma | w_i | w_j, \beta, A \Rightarrow \sigma | w_i, \beta, A \cup \{r(w_i, w_j)\}$

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



# Arc-standard transition-based parser

(there are other transition schemes ...)

Analysis of “I ate fish”

Start



Shift



Shift



**Start:**  $\sigma = [\text{ROOT}], \beta = w_1, \dots, w_n, A = \emptyset$

- 1. Shift  $\sigma, w_i | \beta, A \rightarrow \sigma | w_i, \beta, A$
- 2. Left-Arc<sub>r</sub>  $\sigma | w_i | w_j, \beta, A \rightarrow \sigma | w_j, \beta, A \cup \{r(w_j, w_i)\}$
- 3. Right-Arc<sub>r</sub>  $\sigma | w_i | w_j, \beta, A \rightarrow \sigma | w_i, \beta, A \cup \{r(w_i, w_j)\}$

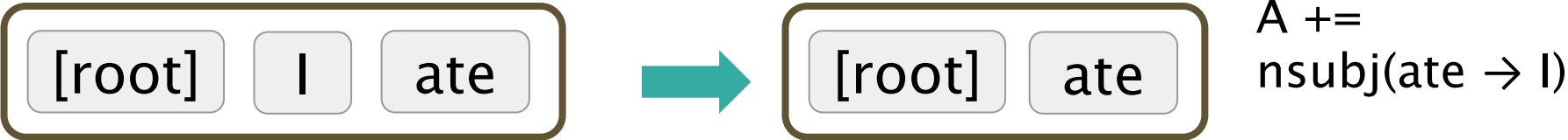
**Finish:**  $\beta = \emptyset$



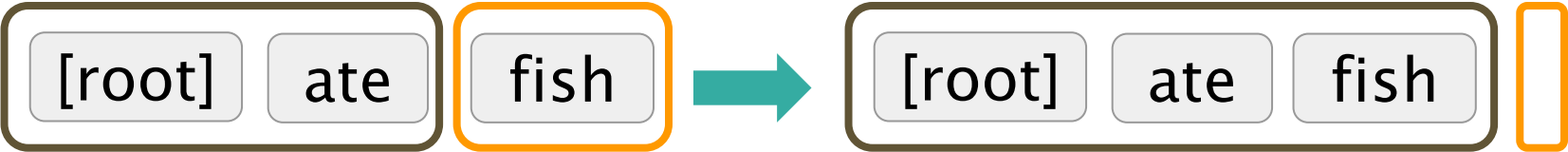
# Arc-standard transition-based parser

## Analysis of “I ate fish”

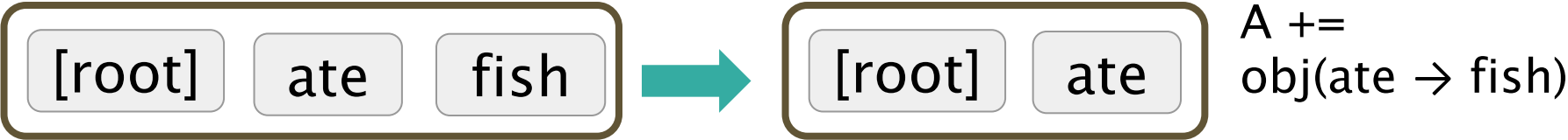
Left Arc



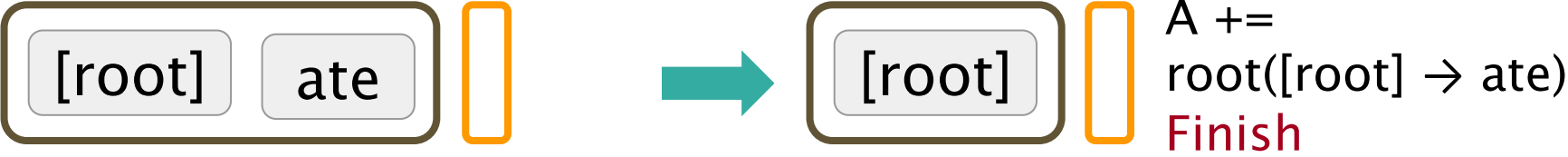
Shift



Right Arc



Right Arc





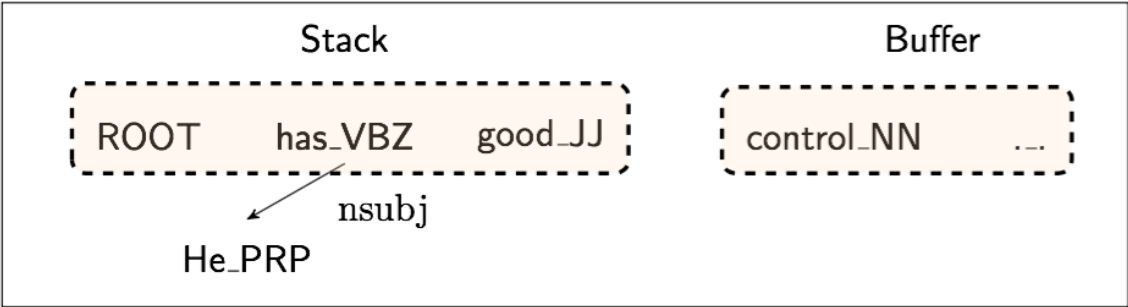
# MaltParser

[Nivre and Hall 2005]

- We have left to explain how we choose the next action
- Each action is predicted by a discriminative classifier (often SVM, can be perceptron, maxent classifier) over each legal move
  - Max of 3 untyped choices; max of  $|R| \times 2 + 1$  when typed
  - Features: top of stack word, POS; first in buffer word, POS; etc.
- There is NO search (in the simplest form)
  - But you can profitably do a beam search if you wish (slower but better)
- It provides **VERY** fast linear time parsing
- The model's accuracy is *slightly* below the best dependency parsers, but
- It provides **fast**, close to state of the art parsing performance



# Feature Representation



binary, sparse  
dim =  $10^6 \sim 10^7$

0	0	0	1	0	0	1	0	...	0	0	1	0
---	---	---	---	---	---	---	---	-----	---	---	---	---

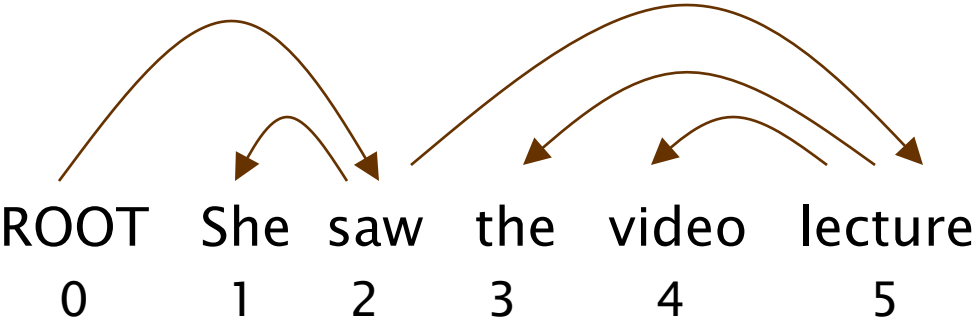
Feature templates: usually a combination of 1 ~ 3 elements from the configuration.

Indicator features

- $s1.w = \text{good} \wedge s1.t = \text{JJ}$
- $s2.w = \text{has} \wedge s2.t = \text{VBZ} \wedge s1.w = \text{good}$
- $lc(s_2).t = \text{PRP} \wedge s2.t = \text{VBZ} \wedge s1.t = \text{JJ}$
- $lc(s_2).w = \text{He} \wedge lc(s_2).l = \text{nsubj} \wedge s2.w = \text{has}$



# Evaluation of Dependency Parsing: (labeled) dependency accuracy



$$\text{Acc} = \frac{\text{\# correct deps}}{\text{\# of deps}}$$

$$\text{UAS} = 4 / 5 = 80\%$$

$$\text{LAS} = 2 / 5 = 40\%$$

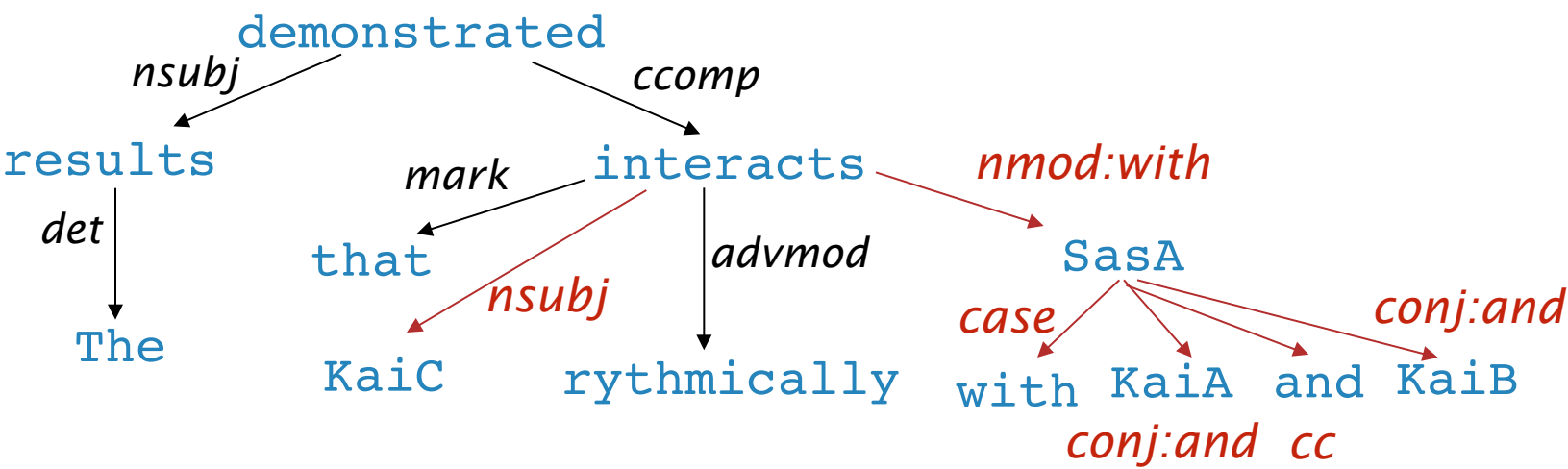
Gold			
1	2	She	nsubj
2	0	saw	root
3	5	the	det
4	5	video	nn
5	2	lecture	obj

Parsed			
1	2	She	nsubj
2	0	saw	root
3	4	the	det
4	5	video	nsubj
5	2	lecture	ccomp



# Dependency paths identify semantic relations – e.g, for protein interaction

[Erkan et al. EMNLP 07, Fundel et al. 2007, etc.]



- KaiC ←*nsubj* interacts *nmod:with* → SasA
- KaiC ←*nsubj* interacts *nmod:with* → SasA *conj:and* → KaiA
- KaiC ←*nsubj* interacts *prep\_with* → SasA *conj:and* → KaiB





# Projectivity

- Dependencies parallel to a CFG tree must be **projective**
  - There must not be any crossing dependency arcs when the words are laid out in their linear order, with all arcs above the words.
- But dependency theory normally does allow non-projective structures to account for displaced constituents
  - You can't easily get the semantics of certain constructions right without these nonprojective dependencies





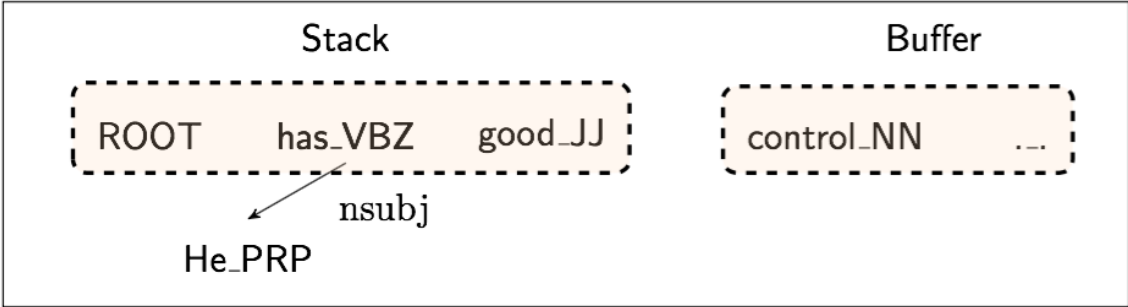
# Handling non-projectivity

- The arc-standard algorithm we presented only builds projective dependency trees
- Possible directions to head:
  1. Just declare defeat on nonprojective arcs
  2. Use a dependency formalism which only admits projective representations (a CFG doesn't represent such structures...)
  3. Use a postprocessor to a projective dependency parsing algorithm to identify and resolve nonprojective links
  4. Add extra transitions that can model at least most non-projective structures (e.g., add an extra SWAP transition, cf. bubble sort)
  5. Move to a parsing mechanism that does not use or require any constraints on projectivity (e.g., the graph-based MSTParser)



# 5. Why train a neural dependency parser? Indicator Features Revisited

- Problem #1
- Problem #2
- Problem #3



dense

dim  $\approx 1000$

0.1	0.9	-0.2	0.3	...	-0.1	-0.5
-----	-----	------	-----	-----	------	------

More than 95% of parsing time is consumed by feature computation.

## Our Approach

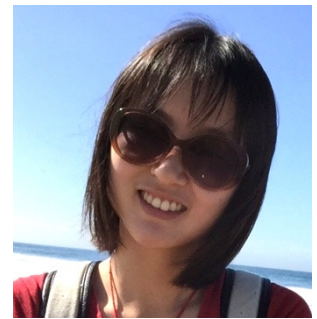
$s1.w = \text{good} \wedge s1.t = \text{JJ}$   
 $s2.w = \text{has} \wedge s2.t = \text{VBZ} \wedge s1.w = \text{good}$   
 $lc(s2).t = \text{PRP} \wedge s2.t = \text{VBZ} \wedge s1.t = \text{JJ}$   
 $lc(s2).w = \text{He} \wedge lc(s2).l = \text{nsubj} \wedge s2.w = \text{has}$

learn a dense and compact feature representation



# A neural dependency parser

## [Chen and Manning 2014]



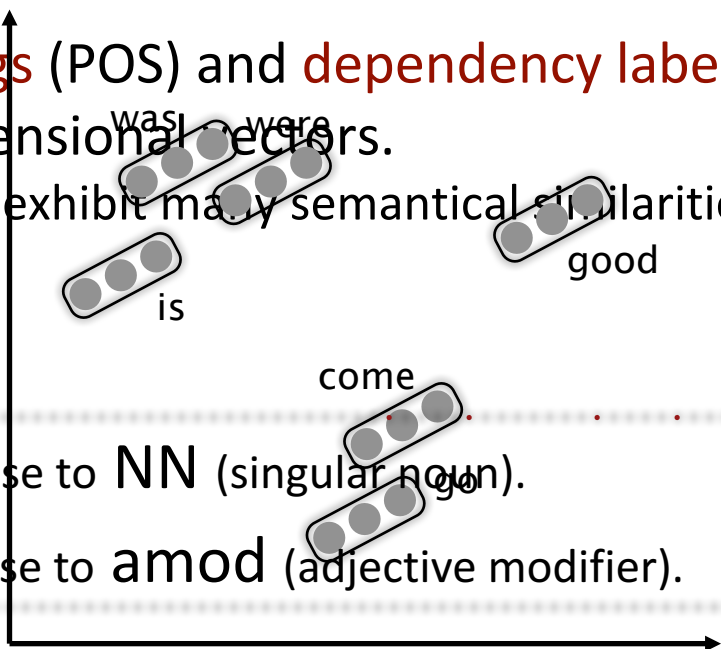
- English parsing to Stanford Dependencies:
  - Unlabeled attachment score (UAS) = head
  - Labeled attachment score (LAS) = head and label

Parser	UAS	LAS	sent. / s
MaltParser	89.8	87.2	469
MSTParser	91.4	88.1	10
TurboParser	<b>92.3*</b>	89.6*	8
C & M 2014	92.0	<b>89.7</b>	<b>654</b>



# Distributed Representations

- We represent each word as a  $d$ -dimensional dense vector (i.e., word embedding)
  - Similar words are expected to have close vectors.
- Meanwhile, **part-of-speech tags** (POS) and **dependency labels** are also represented as  $d$ -dimensional vectors.
  - The smaller discrete sets also exhibit many semantical similarities.

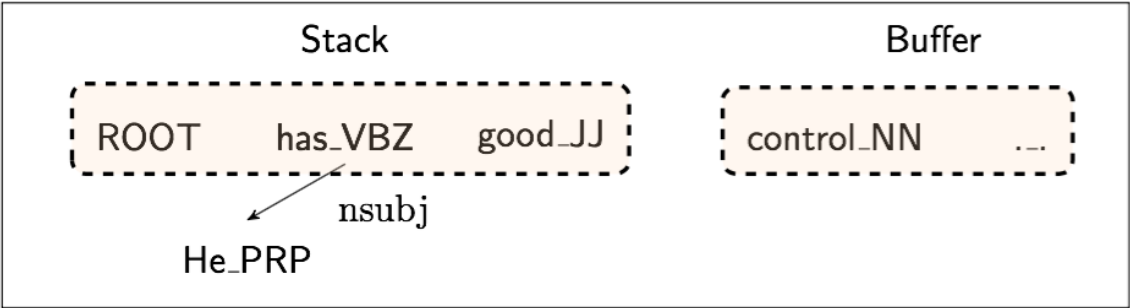


**NNS** (plural noun) should be close to **NN** (singular noun).  
**num** (numerical modifier) should be close to **amod** (adjective modifier).



# Extracting Tokens and then vector representations from configuration

- We extract a set of tokens based on the stack / buffer positions:



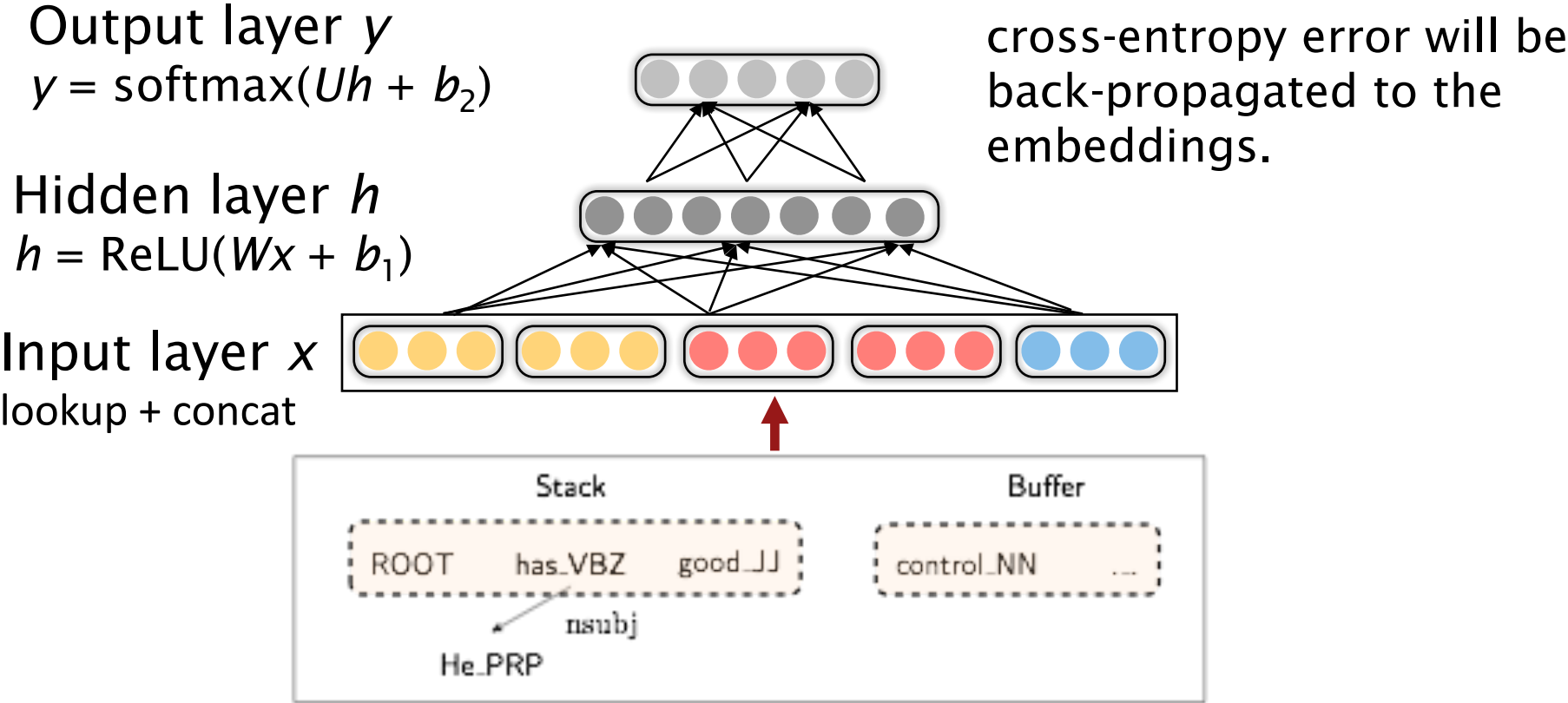
	word	POS	dep.
s <sub>1</sub>	good	JJ	∅
s <sub>2</sub>	has	VBZ	∅
b <sub>1</sub>	control	NN	∅
lc(s <sub>1</sub> )	∅	+	+
rc(s <sub>1</sub> )	∅	∅	∅
lc(s <sub>2</sub> )	He	PRP	nsubj
rc(s <sub>2</sub> )	∅	∅	∅

- We convert them to vector embeddings and concatenate them



# Model Architecture

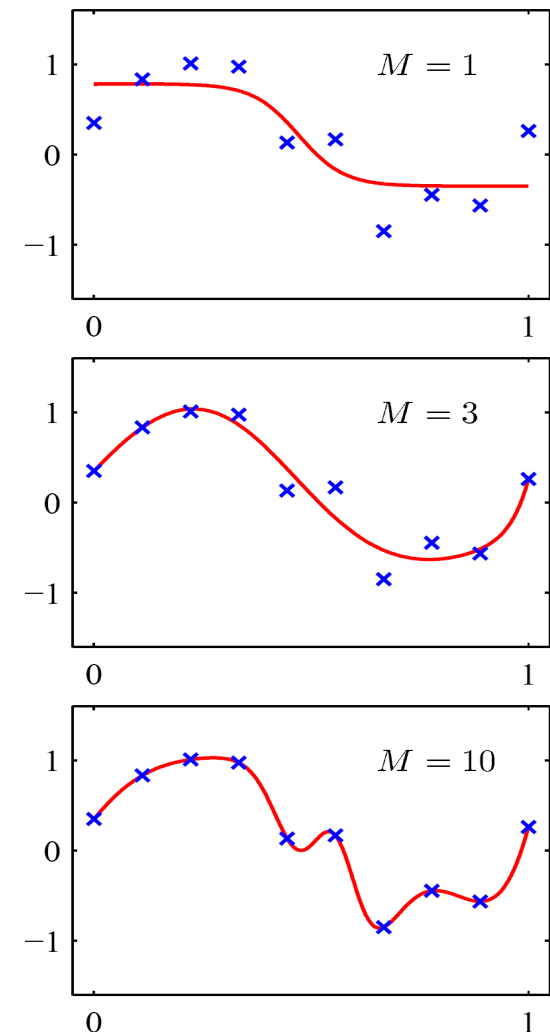
## Softmax probabilities





# Non-linearities between layers: Why they're needed

- For logistic regression: map to probabilities
- Here: function approximation, e.g., for regression or classification
  - Without non-linearities, deep neural networks can't do anything more than a linear transform
    - Extra layers could just be compiled down into a single linear transform
- People use various non-linearities



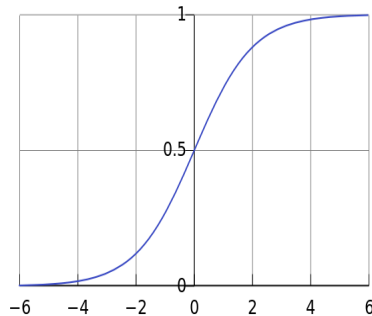




# Non-linearities: What's used

logistic (“sigmoid”)

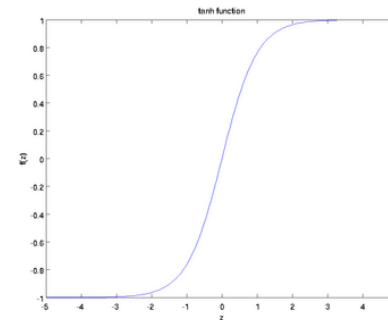
$$f(z) = \frac{1}{1 + \exp(-z)}.$$



$$f'(z) = f(z)(1 - f(z))$$

tanh

$$f(z) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}},$$



$$f'(z) = 1 - f(z)^2$$

tanh is just a rescaled and shifted sigmoid  $\tanh(z) = 2\text{logistic}(2z) - 1$

tanh is often used and often performs better for deep nets

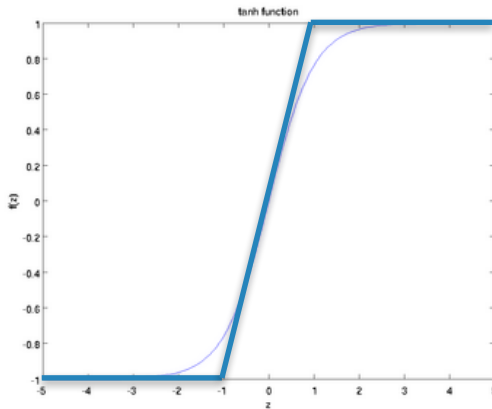
- It's output is symmetric around 0



# Non-linearities: What's used

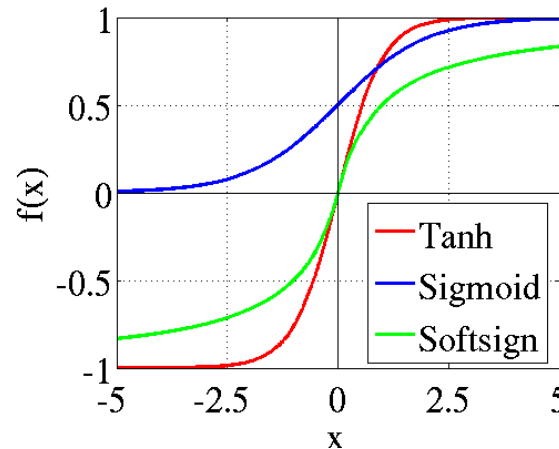
hard tanh

$$\text{HardTanh}(x) = \begin{cases} -1 & \text{if } x < -1 \\ x & \text{if } -1 \leq x \leq 1 \\ 1 & \text{if } x > 1 \end{cases}$$



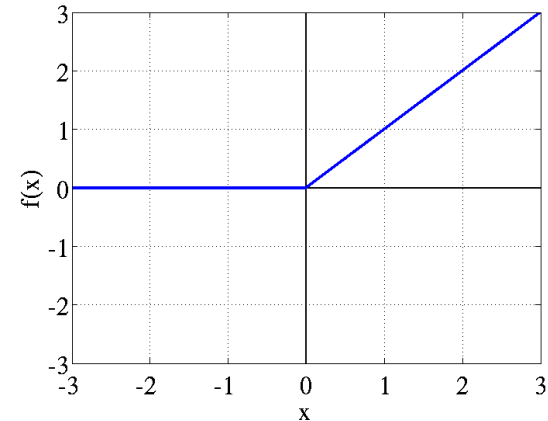
soft sign

$$\text{softsign}(z) = \frac{a}{1 + |a|}$$



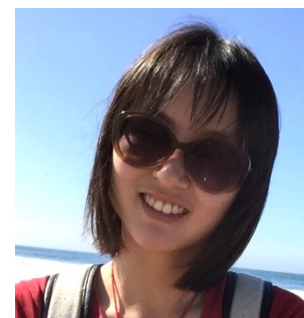
linear rectifier (ReLU)

$$\text{rect}(z) = \max(z, 0)$$

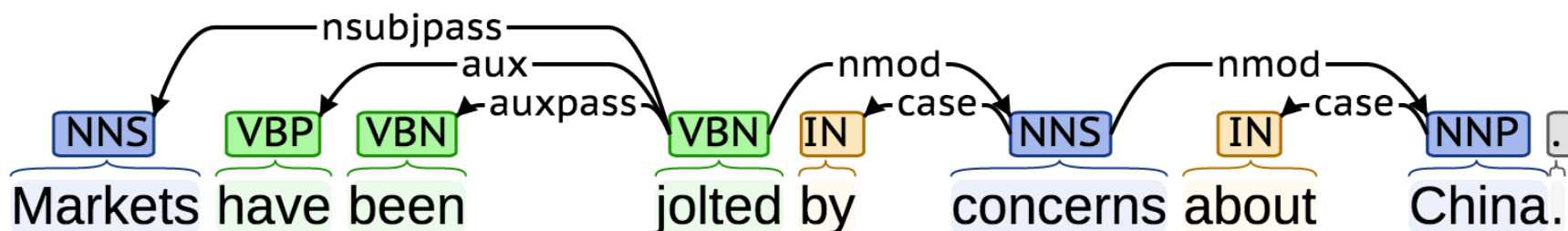


- hard tanh similar but computationally cheaper than tanh and saturates hard.
- [Glorot and Bengio *AISTATS* 2010, 2011] discuss softsign and rectifier
- **Rectified linear unit** is now mega common – transfers linear signal if active

# Dependency parsing for sentence structure



Neural networks can accurately determine the structure of sentences, supporting interpretation



Chen and Manning (2014) was the first simple, successful neural dependency parser

The dense representations let it outperform other greedy parsers in both accuracy and speed

# Further developments in transition-based neural dependency parsing

This work was further developed and improved by others, including in particular at Google

- Bigger, deeper networks with better tuned hyperparameters
- Beam search
- Global, conditional random field (CRF)-style inference over the decision sequence

Leading to SyntaxNet and the Parsey McParseFace model

<https://research.googleblog.com/2016/05/announcing-syntaxnet-worlds-most.html>

Method	UAS	LAS (PTB WSJ SD 3.3)
Chen & Manning 2014	92.0	89.7
Weiss et al. 2015	93.99	92.05
Andor et al. 2016	94.61	92.79