# Natural Language Processing with Deep Learning

# CS224N/Ling284

Christopher Manning

Lecture 18: Tree Recursive Neural Networks, Constituency Parsing, and Sentiment

# Lecture Plan:

Lecture 18: Tree Recursive Neural Networks, Constituency Parsing, and Sentiment

1. Motivation: Compositionality and Recursion (10 mins)
2. Structure prediction with simple Tree RNN: Parsing (20 mins)
3. Backpropagation through Structure (5 mins)
4. More complex TreeRNN units (35 mins)
5. Other uses of tree-recursive neural nets (5 mins)
6. Institute for Human-Centered Artificial Intelligence (5 mins)

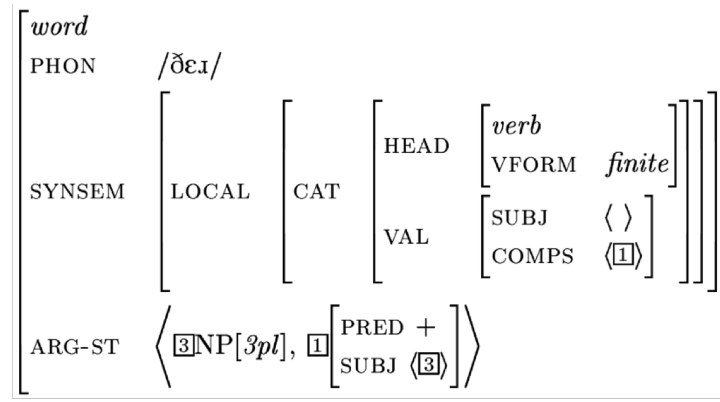
# Last minute project tips

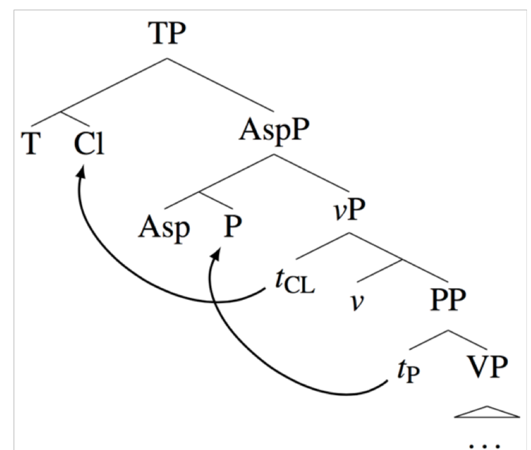
- Nothing works and everything is too slow → Panic

- Simplify model → Go back to basics: bag of vectors + nnet
- Make a very small network and/or dataset for debugging
- Once no bugs: increase model size
- Make sure you can overfit to your training dataset
- Plot your training and dev errors over training iterations
- Once its working, then regularize with L2 and Dropout
- Then if you have time, do some hyperparameter search

- Talk to us in office hours!

# 1. The spectrum of language in CS



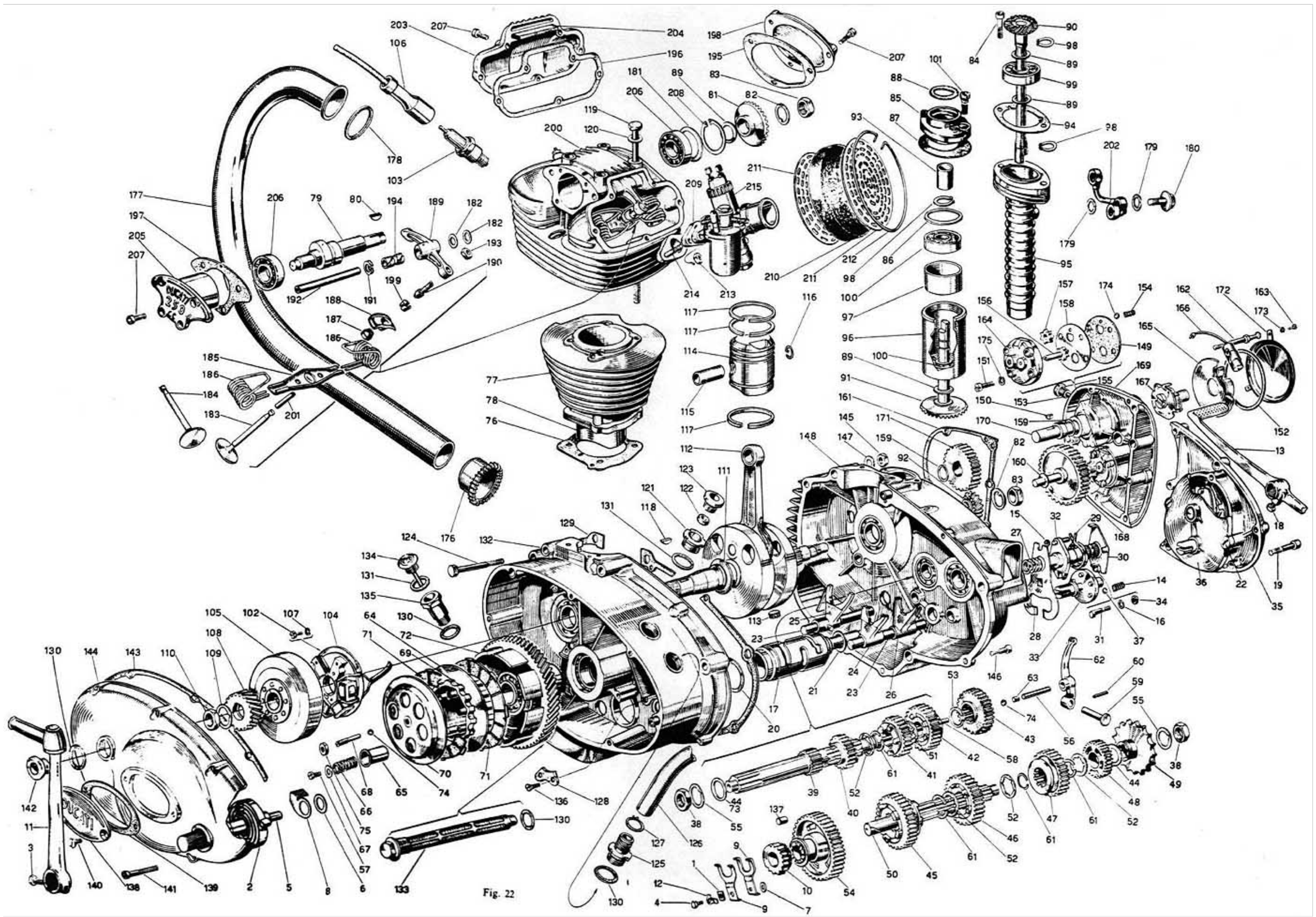
# Semantic interpretation of language – Not just word vectors

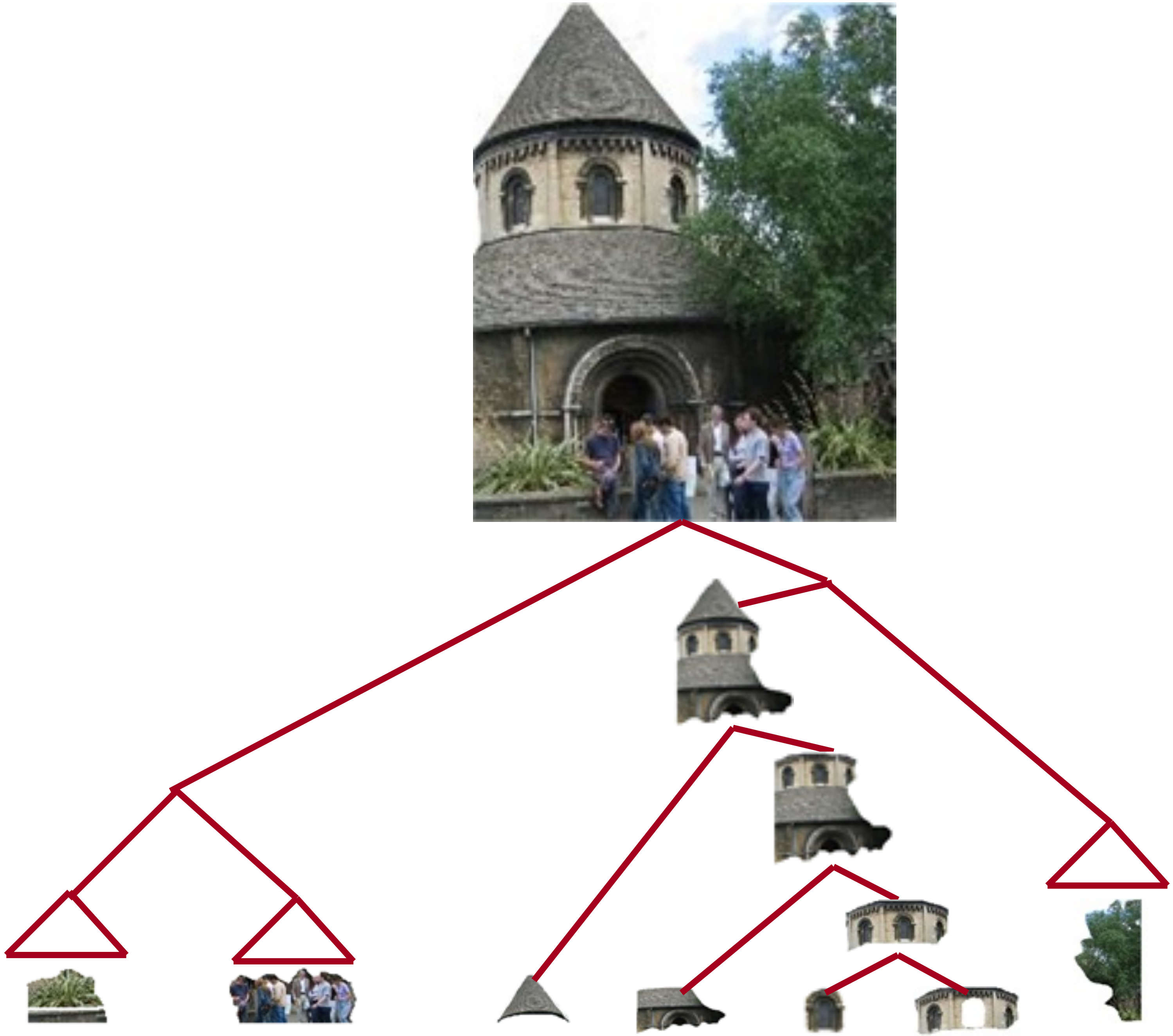
How can we work out the meaning of larger phrases?

- *The **snowboarder** is leaping over a mogul*
- *A **person on a snowboard** jumps into the air*

People interpret the meaning of larger text units – entities, descriptive terms, facts, arguments, stories – by **semantic composition** of smaller elements

# Compositionality





Language understanding –
& Artificial Intelligence – requires being able to understand bigger things from knowing about smaller parts

REVIEW: NEUROSCIENCE

# The Faculty of Language: What Is It, Who Has It, and How Did It Evolve?

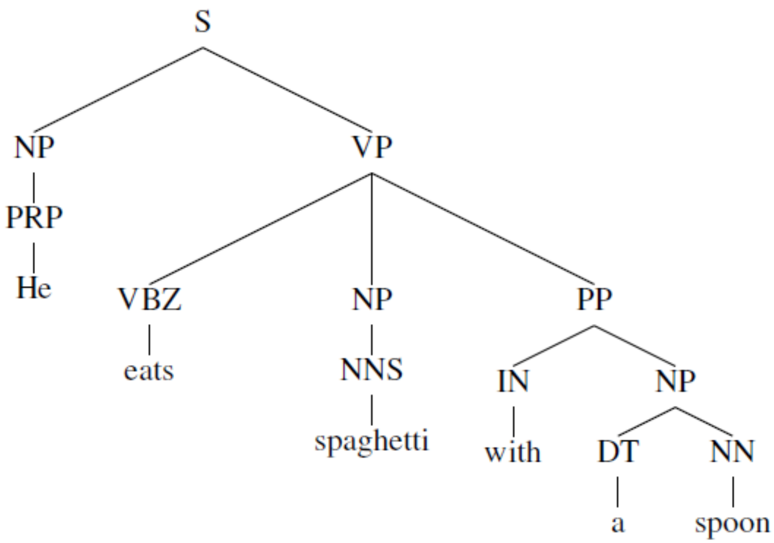
Marc D. Hauser,[1*] Noam Chomsky,[2] W. Tecumseh Fitch[1]

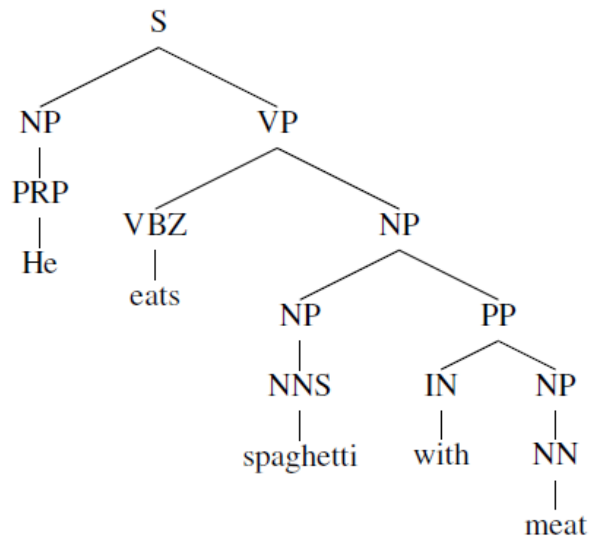
We argue that an understanding of the faculty of language requires substantial interdisciplinary cooperation. We suggest how current developments in linguistics can be profitably wedded to work in evolutionary biology, anthropology, psychology, and neuroscience. We submit that a distinction should be made between the faculty of language in the broad sense (FLB) and in the narrow sense (FLN). FLB includes a sensory-motor system, a conceptual-intentional system, and the computational mechanisms for recursion, providing the capacity to generate an infinite range of expressions from a finite set of elements. We hypothesize that FLN only includes recursion and is the only uniquely human component of the faculty of language. We further argue that FLN may have evolved for reasons other than language, hence comparative studies might look for evidence of such computations outside of the domain of communication (for example, number, navigation, and social relations).

If a martian graced our planet, it would be struck by one remarkable similarity among Earth's living creatures and a key difference. Concerning similarity, it would note that all

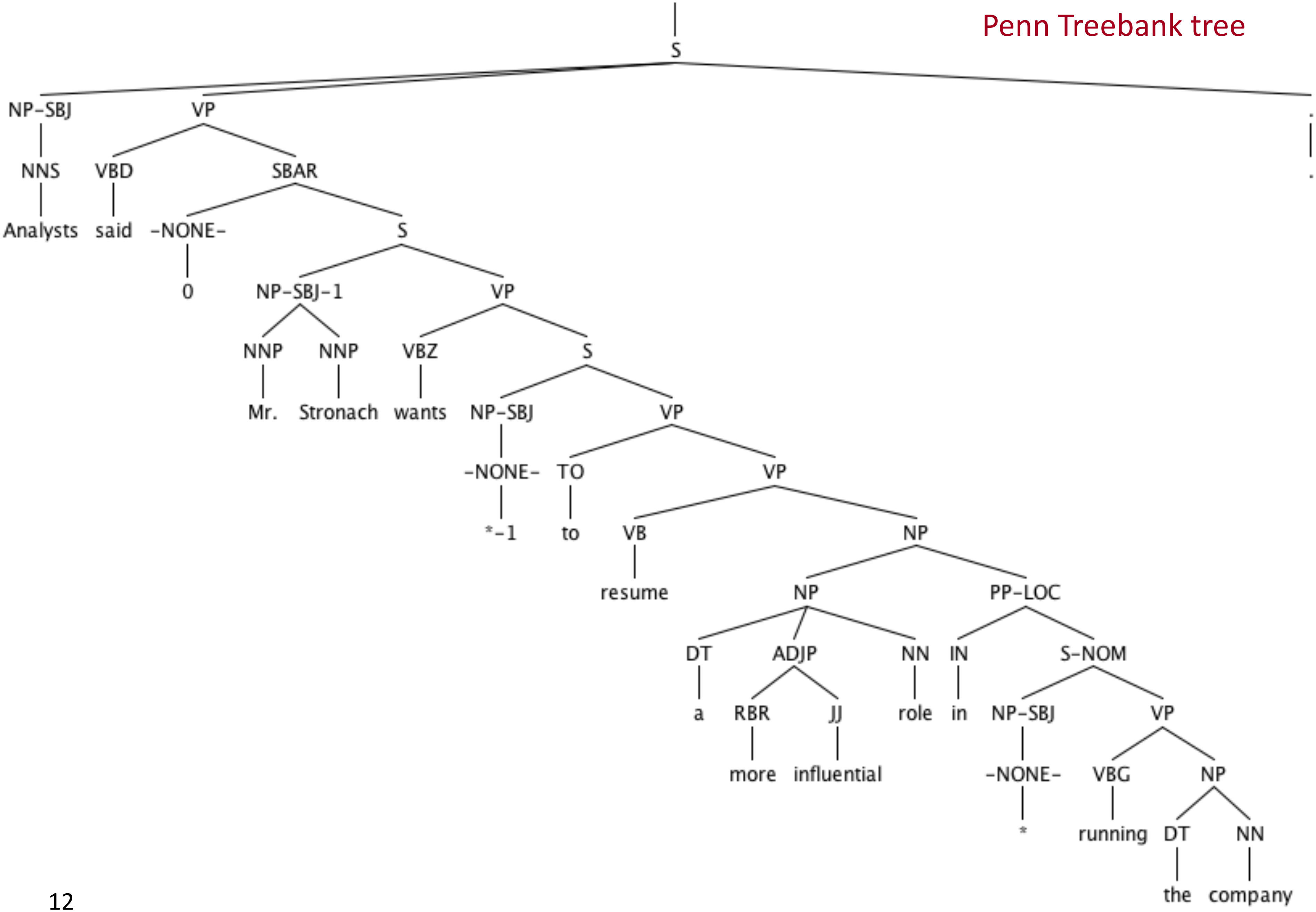
# Are languages recursive?

- Cognitively somewhat debatable (need to head to infinity)
- But: recursion is natural for describing language
  - *[The person standing next to [the man from [the company that purchased [the firm that you used to work at]]]]*
  - noun phrase containing a noun phrase containing a noun phrase
- It's a very powerful prior for language structure

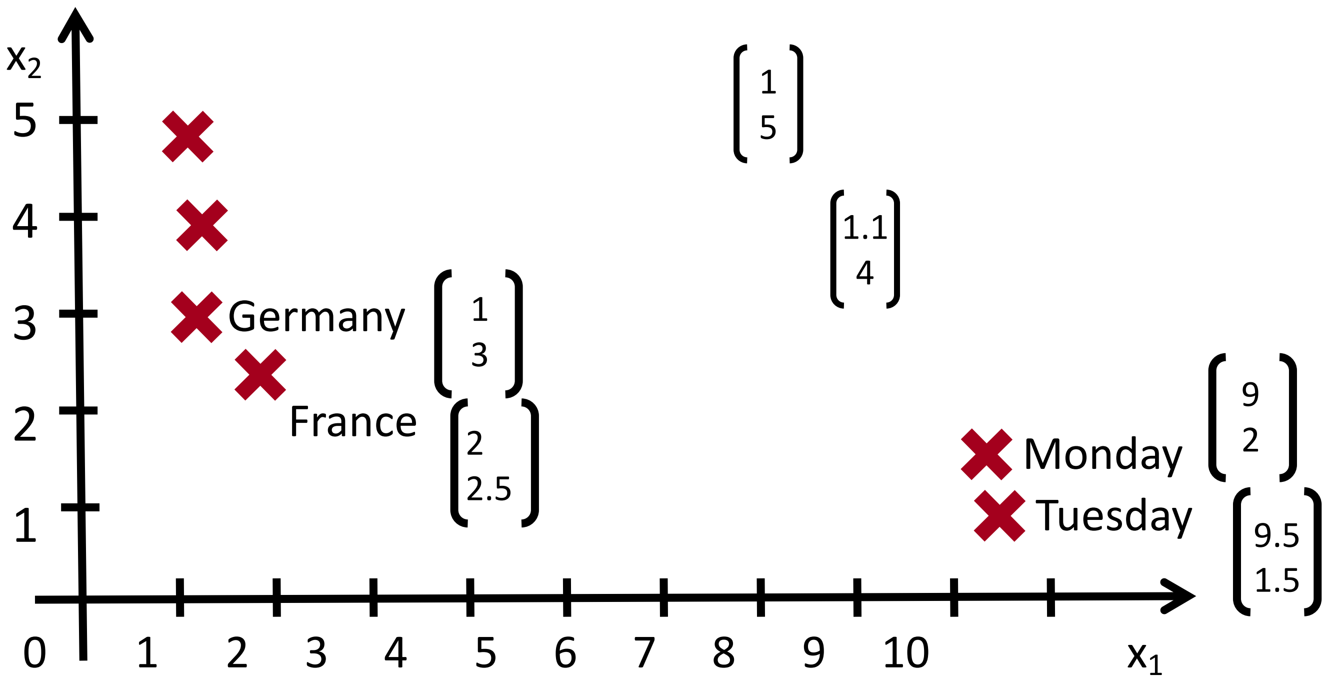


Penn Treebank tree



# 2. Building on Word Vector Space Models




the country of my birth

the place where I was born

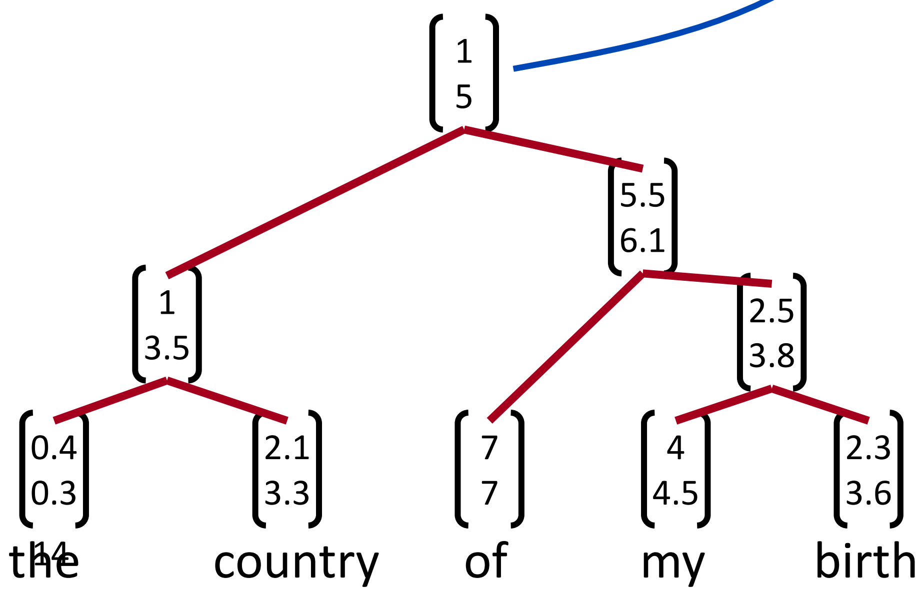
How can we represent the meaning of longer phrases?

By mapping them into the same vector space!

# How should we map phrases into a vector space?

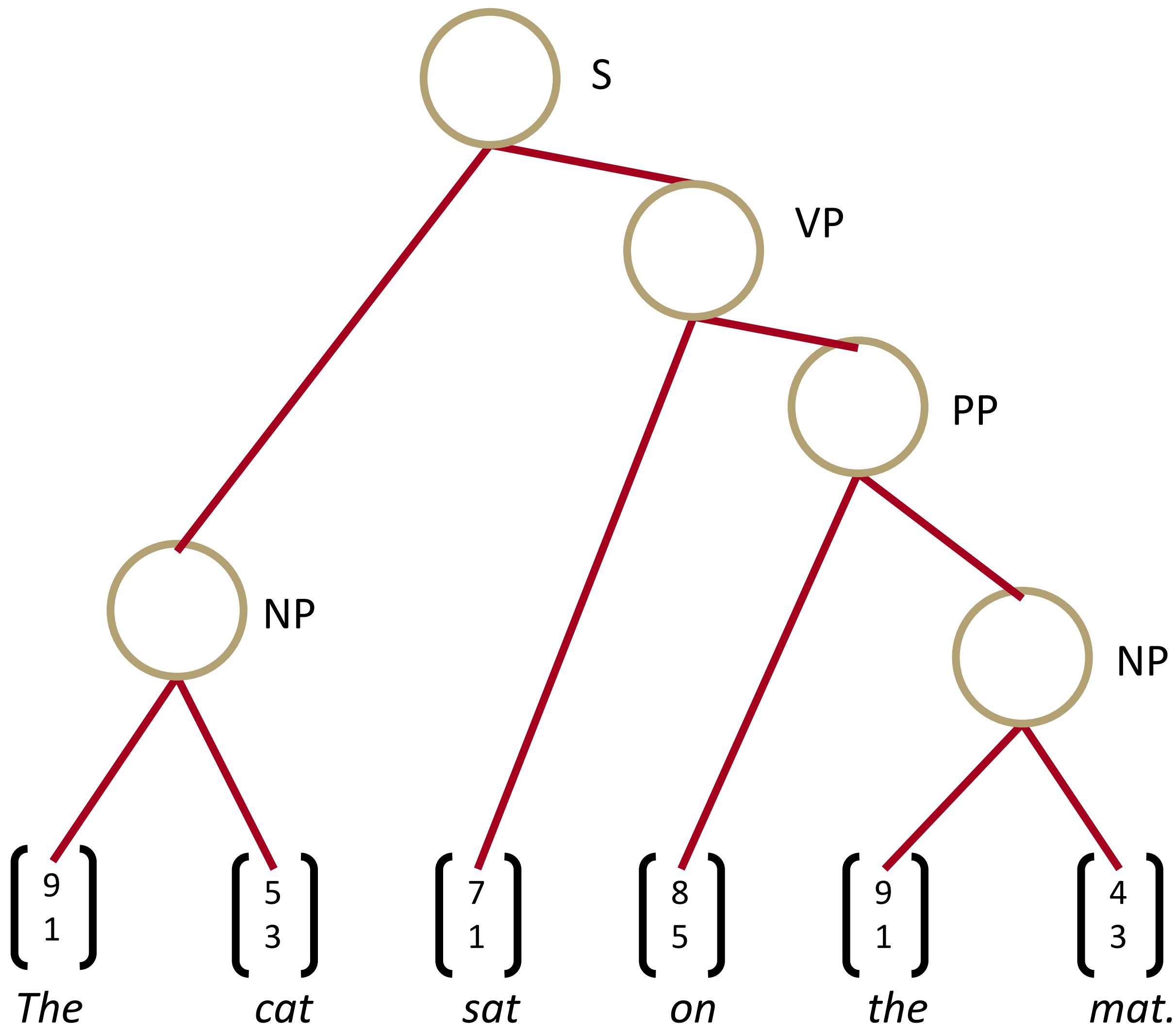
Socher, Manning, and Ng. ICML, 2011

| Use principle of compositionality |
|---|
| The meaning (vector) of a sentence is determined by<br>(1) the meanings of its words and<br>(2) the rules that combine them. |

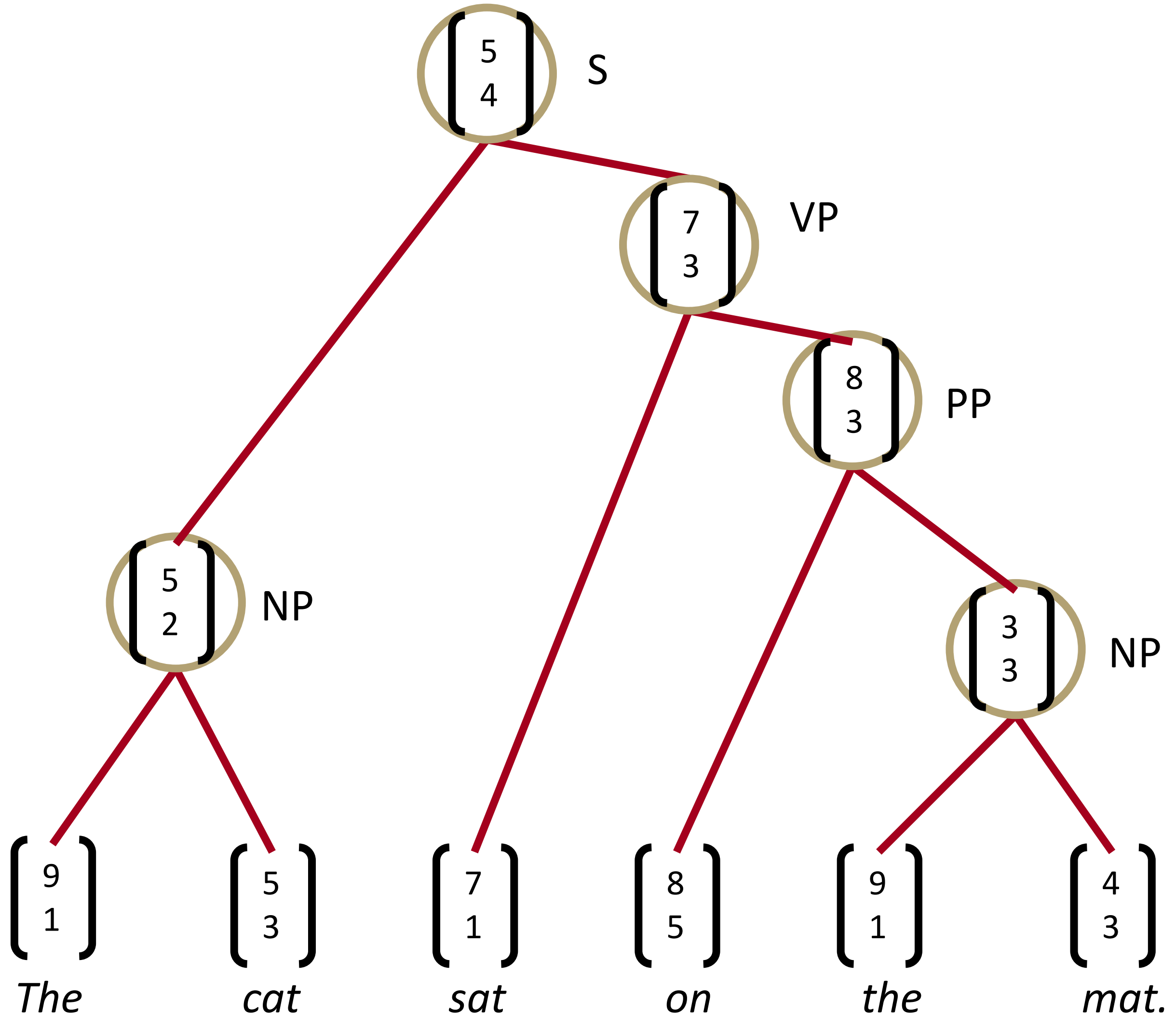




Models in this section can jointly learn parse trees and compositional vector representations

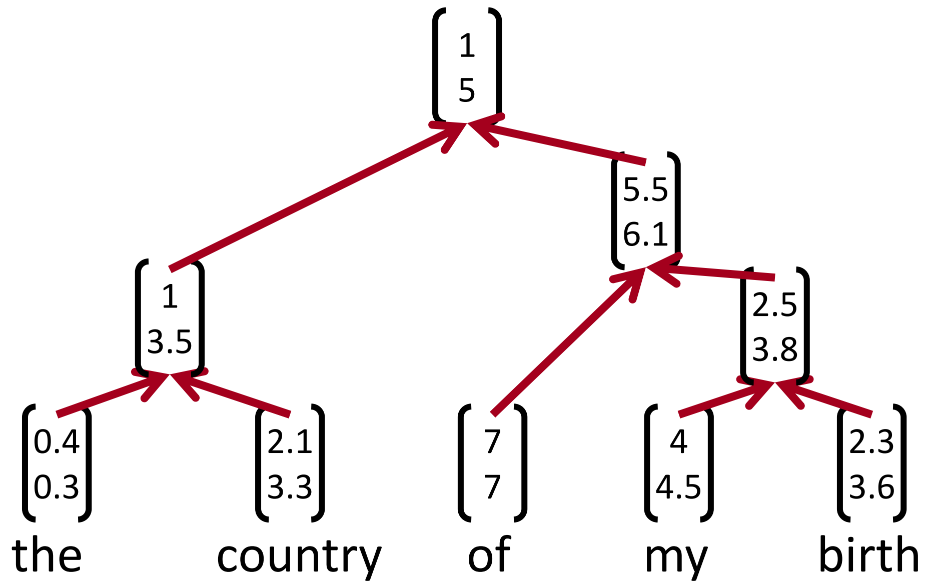
# Constituency Sentence Parsing: What we want

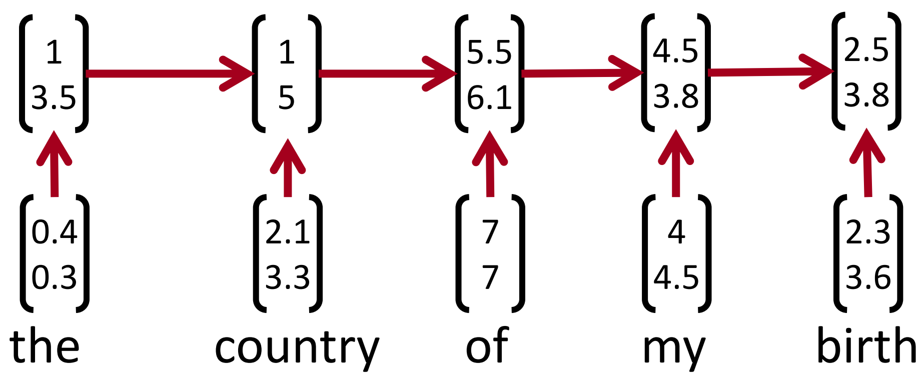


# Learn Structure and Representation



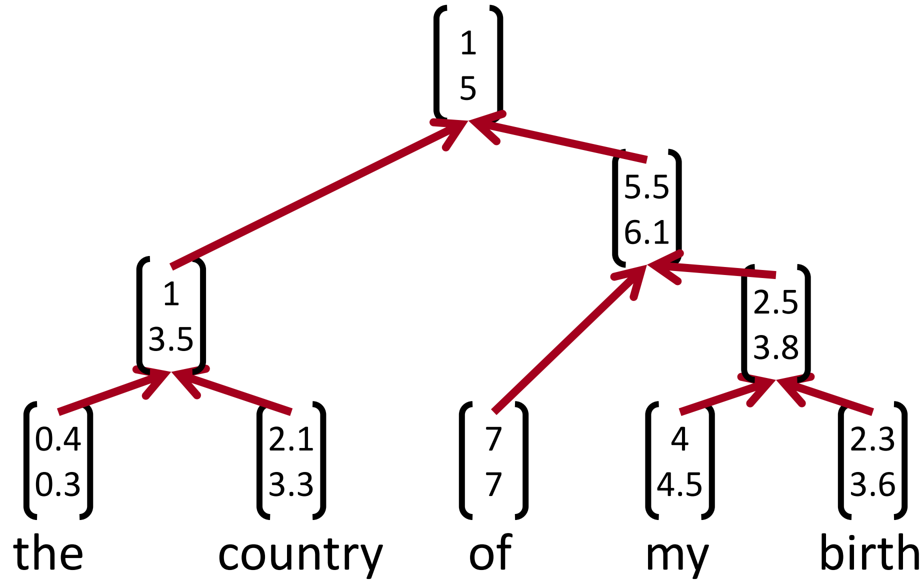
# Recursive vs. recurrent neural networks



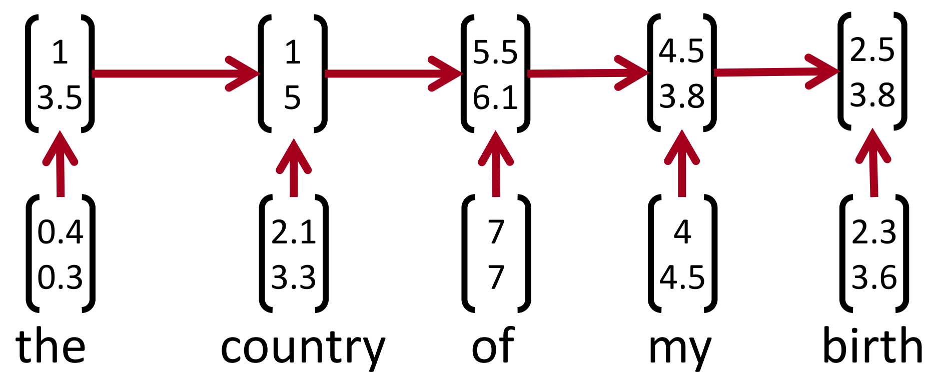



# Recursive vs. recurrent neural networks

- Recursive neural nets require a tree structure



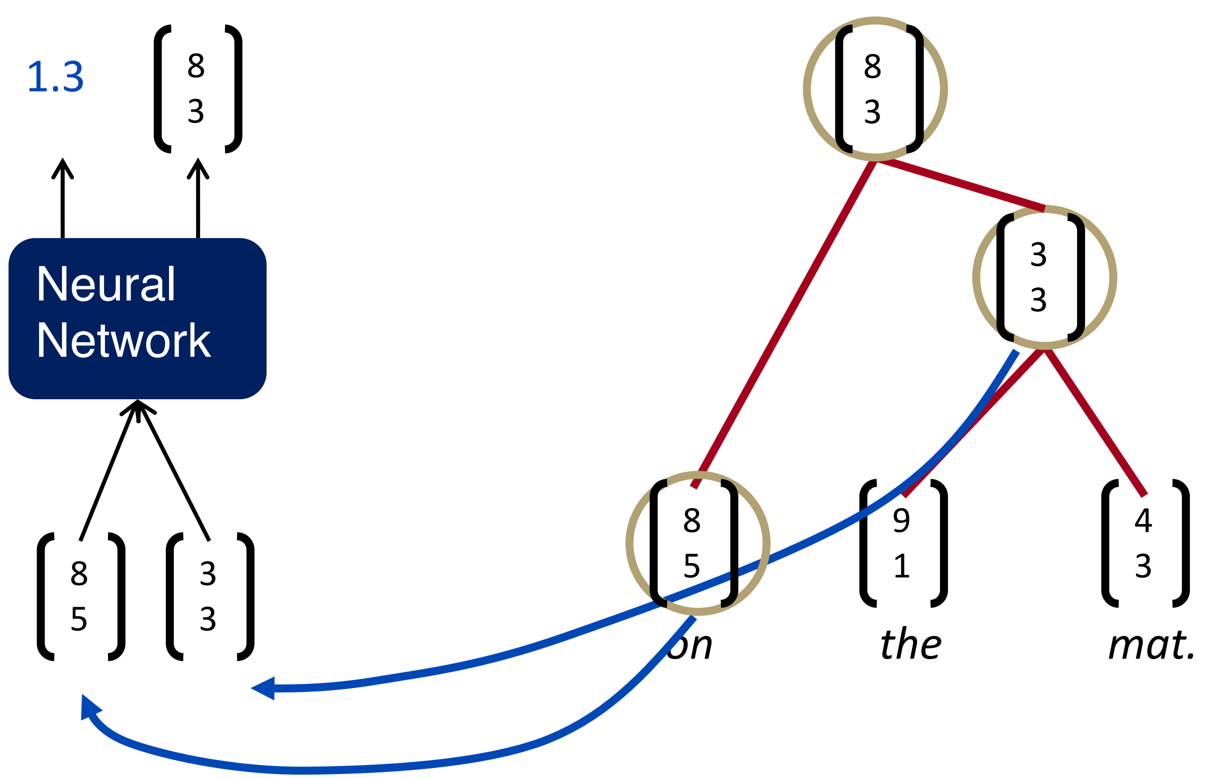

- Recurrent neural nets cannot capture phrases without prefix context and often capture too much of last words in final vector



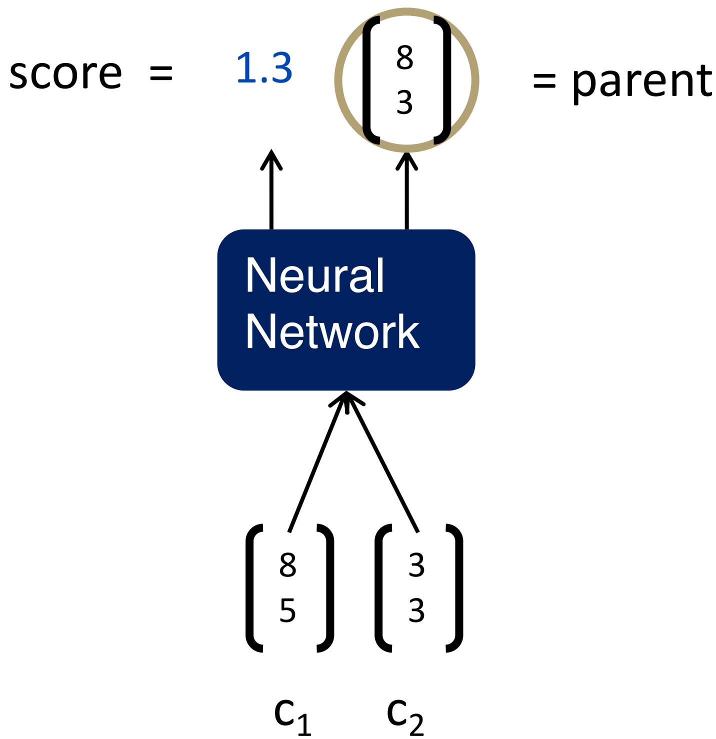
# Recursive Neural Networks for Structure Prediction

Inputs: two candidate children's representations
Outputs:

1. The semantic representation if the two nodes are merged.
2. Score of how plausible the new node would be.



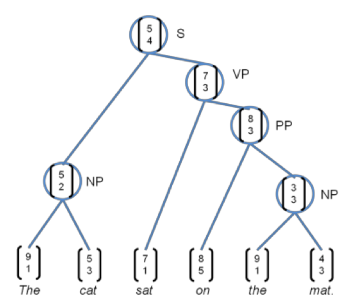
# Recursive Neural Network Definition



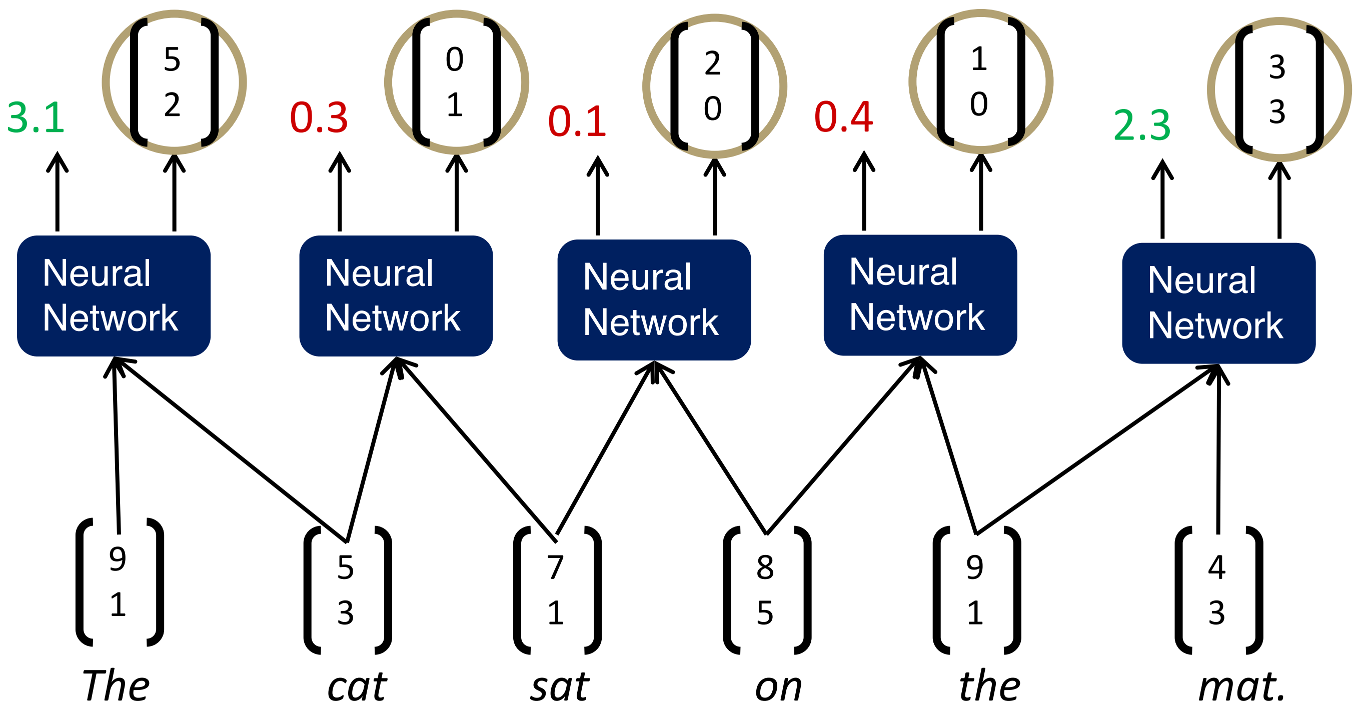

$$\text{score} = U^{\mathsf{T}} p$$

$$\boxed{p = \tanh\left(W \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} + b\right),}$$

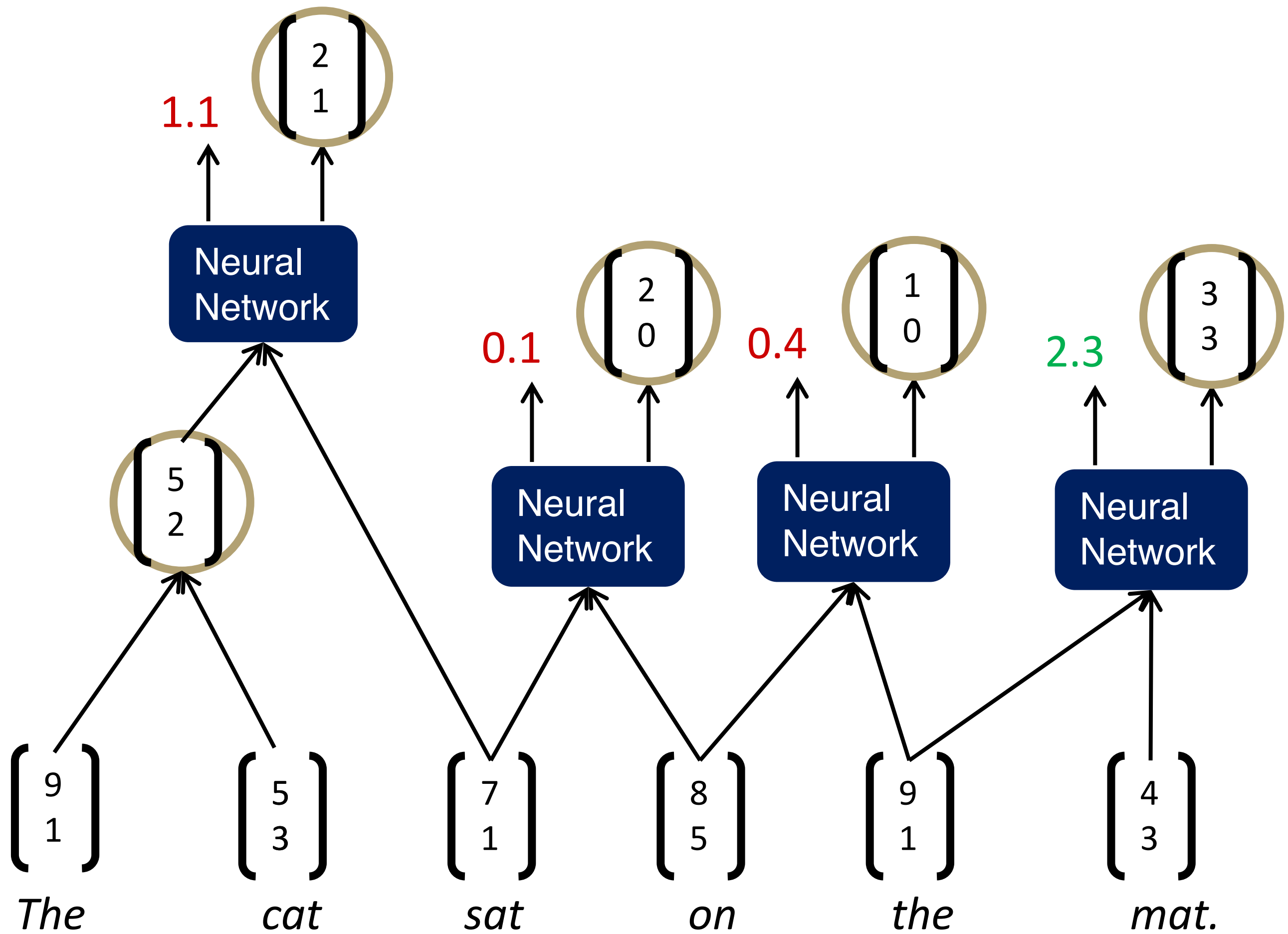
**Same** *W* parameters at all nodes of the tree



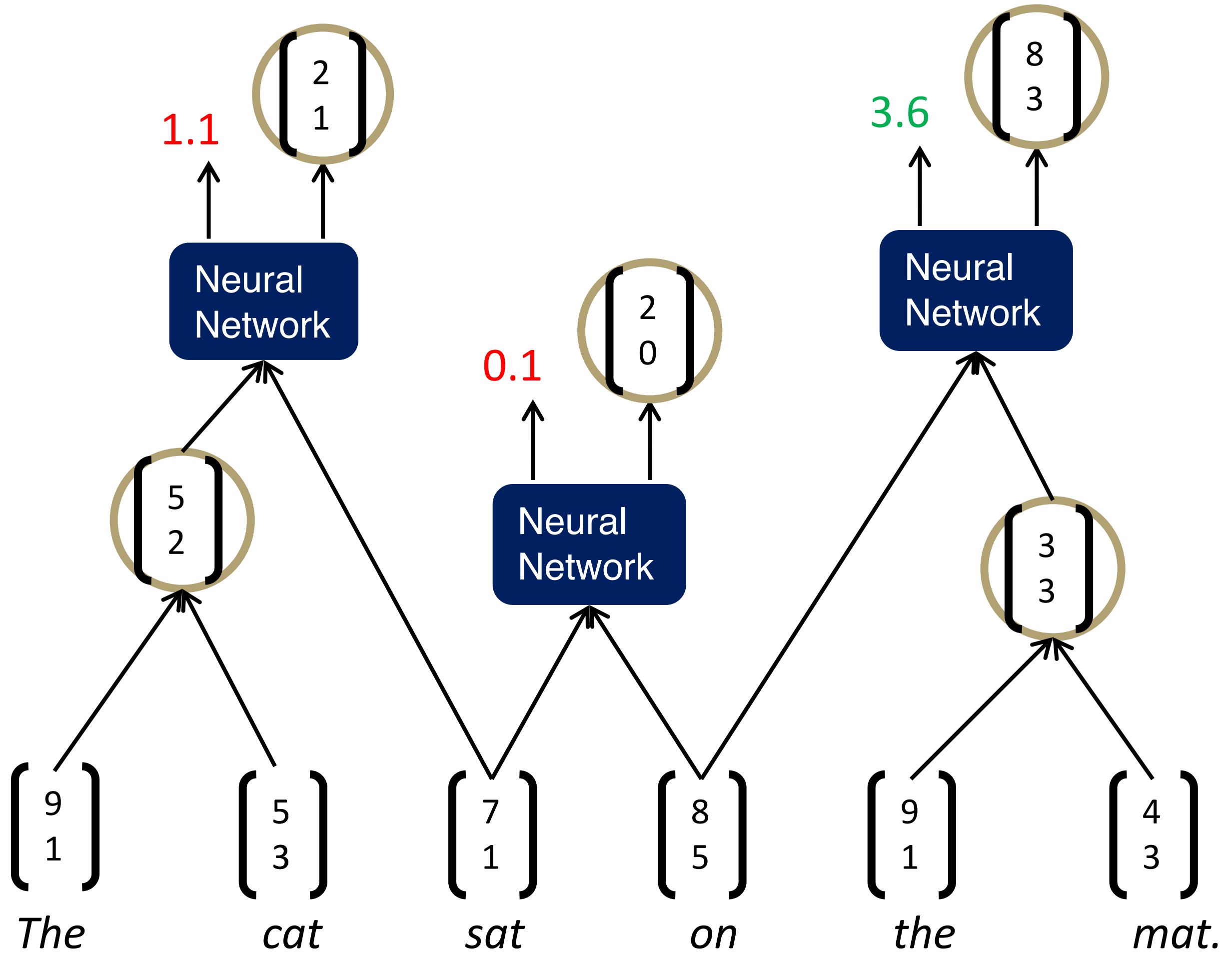
# Parsing a sentence with an RNN (greedily)



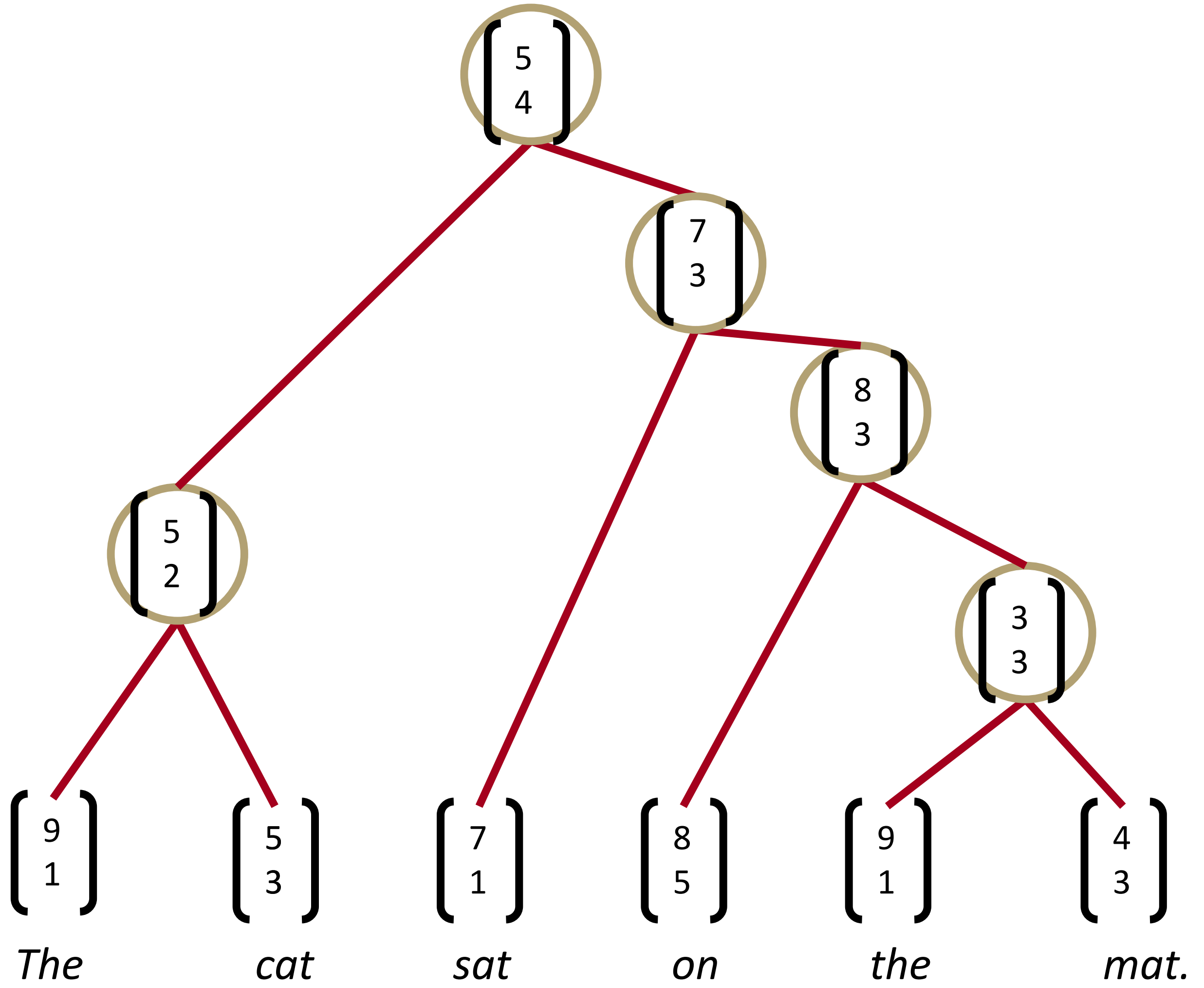
# Parsing a sentence



# Parsing a sentence

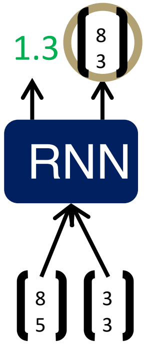


# Parsing a sentence



# Max-Margin Framework - Details

- The score of a tree is computed by the sum of the parsing decision scores at each node:




$$s(x, y) = \sum_{n \in nodes(y)} s_n$$

- *x* is sentence; *y* is parse tree

# Max-Margin Framework - Details

- Similar to max-margin parsing (Taskar et al. 2004), a supervised max-margin objective

$$J = \sum_i s(x_i, y_i) - \max_{y \in A(x_i)} \left( s(x_i, y) + \Delta(y, y_i) \right)$$

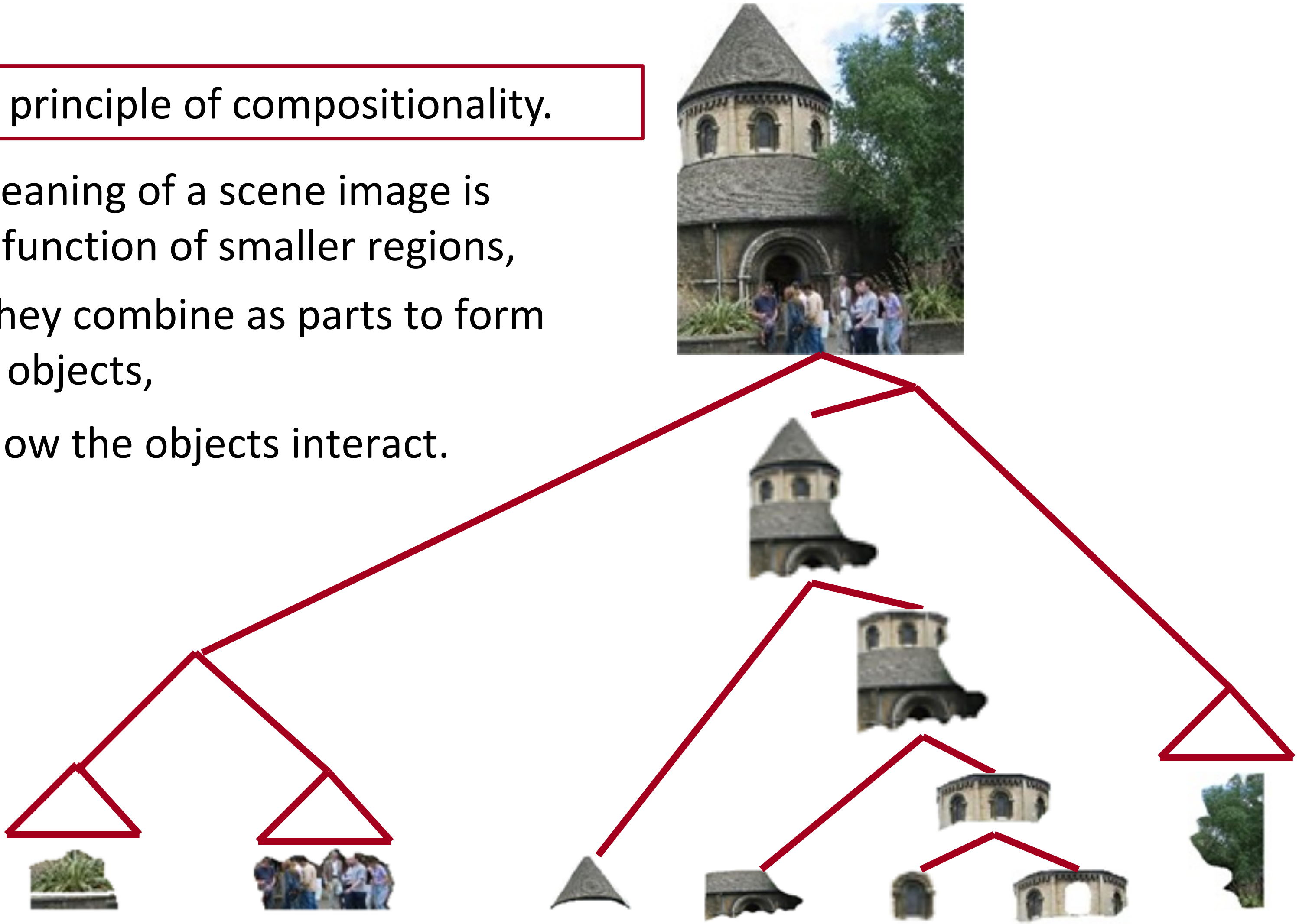
- The loss $\Delta(y, y_i)$ penalizes all incorrect decisions

- Structure search for A(x) was greedy (join best nodes each time)
  - Instead: Beam search with chart

# Scene Parsing

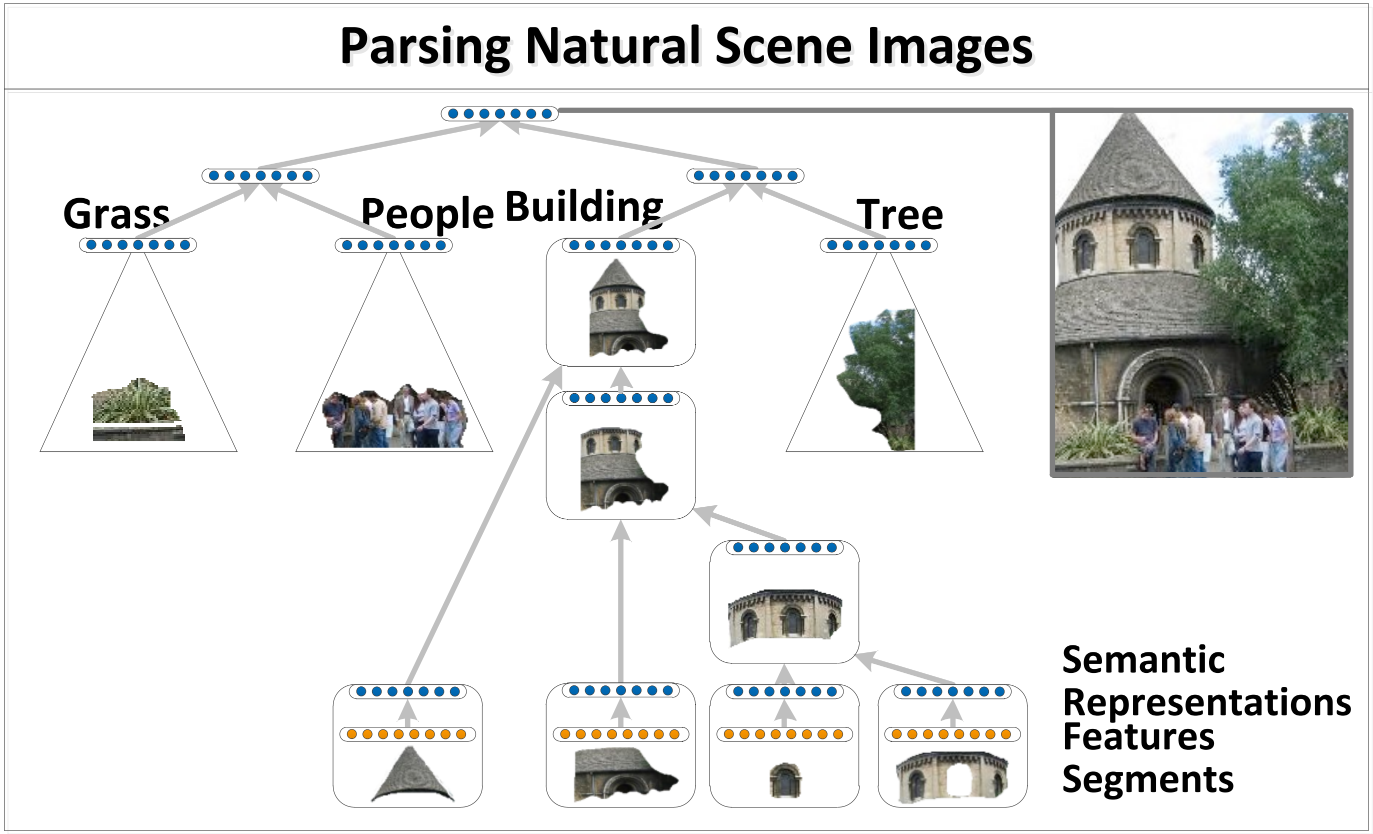
Similar principle of compositionality.

- The meaning of a scene image is also a function of smaller regions,
- how they combine as parts to form larger objects,
- and how the objects interact.

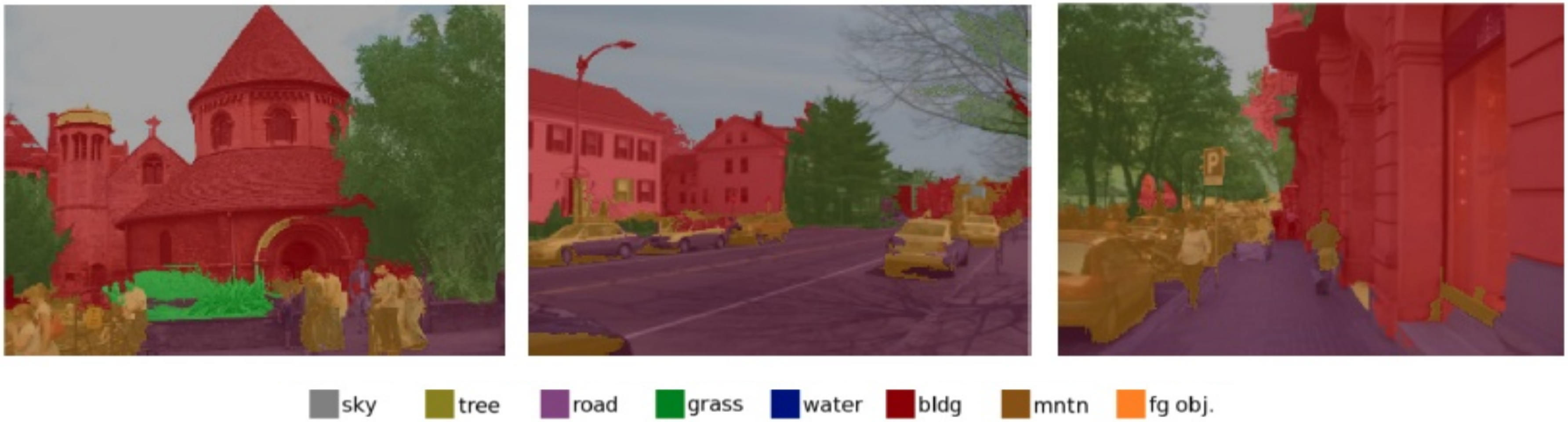


# Algorithm for Parsing Images

Same Recursive Neural Network as for natural language parsing!
(Socher et al. ICML 2011)



# Multi-class segmentation




| Method | Accuracy |
|---|---|
| Pixel CRF (Gould et al., ICCV 2009) | 74.3 |
| Classifier on superpixel features | 75.9 |
| Region-based energy (Gould et al., ICCV 2009) | 76.4 |
| Local labelling (Tighe & Lazebnik, ECCV 2010) | 76.9 |
| Superpixel MRF (Tighe & Lazebnik, ECCV 2010) | 77.5 |
| Simultaneous MRF (Tighe & Lazebnik, ECCV 2010) | 77.5 |
| Recursive Neural Network | **78.1** |

Stanford Background Dataset (Gould et al. 2009)

# 3. Backpropagation Through Structure

Introduced by Goller & Küchler (1996)

Principally the same as general backpropagation

$$\boxed{\delta^{(l)} = \left((W^{(l)})^T \delta^{(l+1)}\right) \circ f'(z^{(l)}),}$$

$$\boxed{\frac{\partial}{\partial W^{(l)}} E_R = \delta^{(l+1)} (a^{(l)})^T + \lambda W^{(l)}}$$

Calculations resulting from the recursion and tree structure:

1. Sum derivatives of *W* from all nodes (like RNN)
2. Split derivatives at each node (for tree)
3. Add error messages from parent + node itself

# BTS: 1) Sum derivatives of all nodes

You can actually assume it's a different *W* at each node

Intuition via example:

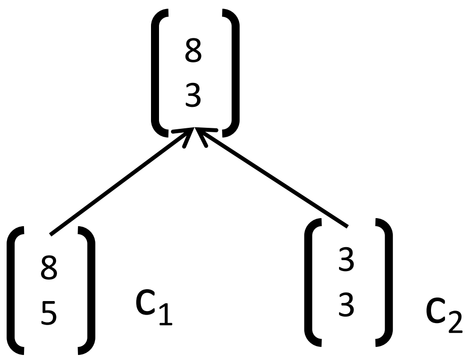
$$
\begin{aligned}
& \frac{\partial}{\partial W} f(W(f(Wx)) \\
= \quad & f'(W(f(Wx)) \left( \left( \frac{\partial}{\partial W} W \right) f(Wx) + W \frac{\partial}{\partial W} f(Wx) \right) \\
= \quad & f'(W(f(Wx)) \left( f(Wx) + W f'(Wx) x \right)
\end{aligned}
$$

If we take separate derivatives of each occurrence, we get same:

$$
\begin{aligned}
& \frac{\partial}{\partial W_2} f(W_2(f(W_1 x)) + \frac{\partial}{\partial W_1} f(W_2(f(W_1 x)) \\
= \quad & f'(W_2(f(W_1 x)) \left( f(W_1 x) \right) + f'(W_2(f(W_1 x)) \left( W_2 f'(W_1 x) x \right) \\
= \quad & f'(W_2(f(W_1 x)) \left( f(W_1 x) + W_2 f'(W_1 x) x \right) \\
= \quad & f'(W(f(Wx)) \left( f(Wx) + W f'(Wx) x \right)
\end{aligned}
$$

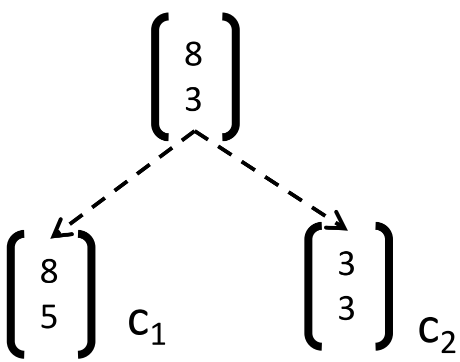
# BTS: 2) Split derivatives at each node

During forward prop, the parent is computed using 2 children




$$p = \tanh\left(W\begin{bmatrix} c_1 \\ c_2 \end{bmatrix} + b\right)$$

Hence, the errors need to be computed wrt each of them:

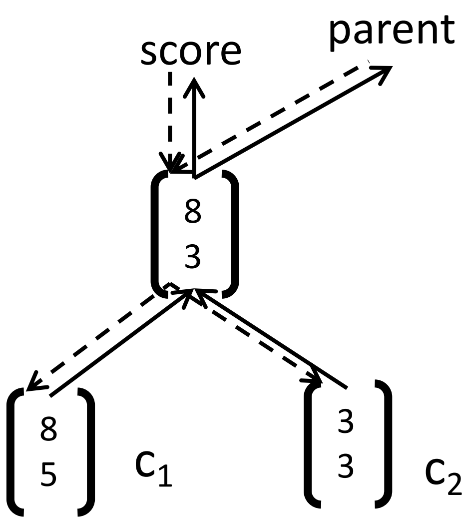



where each child's error is *n*-dimensional

$$\delta_{p \to c_1 c_2} = [\delta_{p \to c_1} \delta_{p \to c_2}]$$

# BTS: 3) Add error messages

- At each node:
  - What came up (fprop) must come down (bprop)
  - Total error messages = error messages from parent + error message from own score



# BTS Python Code: forwardProp

```
def forwardProp(self,node):
    # Recursion
    ...

    # This node's hidden activation
    node.h = np.dot(self.W,np.hstack([node.left.h, node.right.h])) + self.b
    # Relu
    node.h[node.h<0] = 0

    # Softmax
    node.probs = np.dot(self.Ws,node.h) + self.bs
    node.probs -= np.max(node.probs)
    node.probs = np.exp(node.probs)
    node.probs = node.probs/np.sum(node.probs)
```

# BTS Python Code: backProp

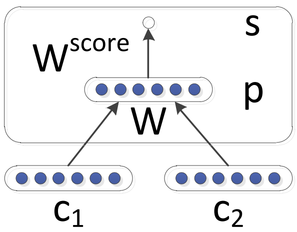
```
def backProp(self,node,error=None):
    # Softmax grad
    deltas = node.probs
    deltas[node.label] -= 1.0
    self.dWs += np.outer(deltas,node.h)
    self.dbs += deltas
    deltas = np.dot(self.Ws.T,deltas)

    # Add deltas from above
    if error is not None:
        deltas += error

    # f'(z) now:
    deltas *= (node.h != 0)

    # Update word vectors if leaf node:
    if node.isLeaf:
        self.dL[node.word] += deltas
        return

    # Recursively backprop
    if not node.isLeaf:
        self.dW += np.outer(deltas,np.hstack([node.left.h, node.right.h]))
        self.db += deltas
        # Error signal to children
        deltas = np.dot(self.W.T, deltas)
        self.backProp(node.left, deltas[:self.hiddenDim])
        self.backProp(node.right, deltas[self.hiddenDim:])
```

$$\boxed{\delta^{(l)} = \left((W^{(l)})^T \delta^{(l+1)}\right) \circ f'(z^{(l)}),}$$

$$\boxed{\frac{\partial}{\partial W^{(l)}} E_R = \delta^{(l+1)} (a^{(l)})^T + \lambda W^{(l)}}$$

# Discussion: Simple TreeRNN

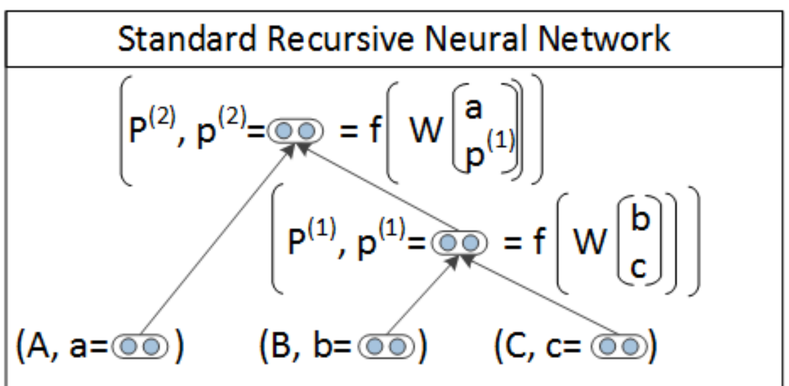
- Decent results with single matrix TreeRNN
- Single weight matrix TreeRNN could capture some phenomena but not adequate for more complex, higher order composition and parsing long sentences
- There is no real interaction between the input words
- The composition function is the same for all syntactic categories, punctuation, etc.

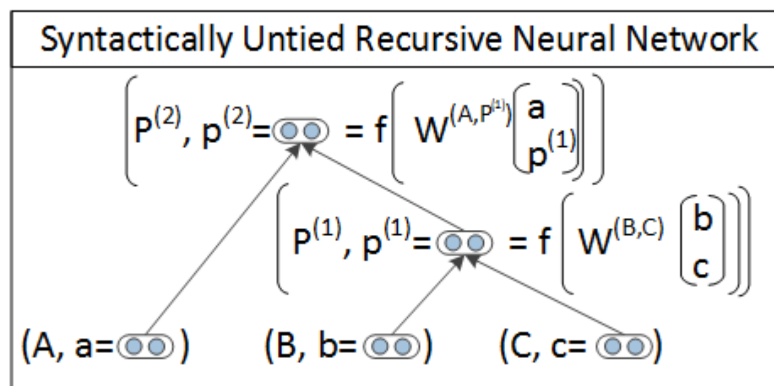


# 4. Version 2: Syntactically-Untied RNN

[Socher, Bauer, Manning, Ng 2013]

- A symbolic Context-Free Grammar (CFG) backbone is adequate for basic syntactic structure
- We use the discrete syntactic categories of the children to choose the composition matrix
- A TreeRNN can do better with different composition matrix for different syntactic environments
- The result gives us a better semantics



# Compositional Vector Grammars

- Problem: Speed. Every candidate score in beam search needs a matrix-vector product.

- Solution: Compute score only for a subset of trees coming from a simpler, faster model (PCFG)
  - Prunes very unlikely candidates for speed
  - Provides coarse syntactic categories of the children for each beam candidate

- Compositional Vector Grammar = PCFG + TreeRNN

# Related Work for parsing

- Resulting CVG Parser is related to previous work that extends PCFG parsers
- Klein and Manning (2003a) : manual feature engineering
- Petrov et al. (2006) : learning algorithm that splits and merges syntactic categories
- Lexicalized parsers (Collins, 2003; Charniak, 2000): describe each category with a lexical item
- Hall and Klein (2012) combine several such annotation schemes in a factored parser.
- CVGs extend these ideas from discrete representations to richer continuous ones

# Experiments

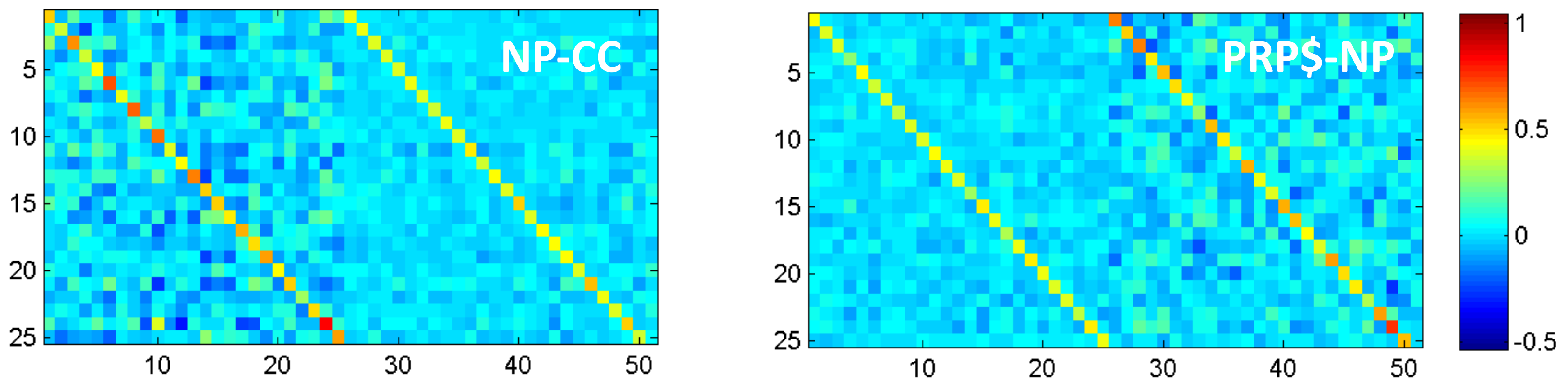
- Standard *WSJ* split, labeled F1
- Based on simple PCFG with fewer states
- Fast pruning of search space, few matrix-vector products
- 3.8% higher F1

| Parser | Test, All Sentences |
|---|---|
| Stanford PCFG, (Klein and Manning, 2003a) | 85.5 |
| Stanford Factored (Klein and Manning, 2003b) | 86.6 |
| Factored PCFGs (Hall and Klein, 2012) | 89.4 |
| Collins (Collins, 1997) | 87.7 |
| SSN (Henderson, 2004) | 89.4 |
| Berkeley Parser (Petrov and Klein, 2007) | 90.1 |
| CVG (RNN) (Socher et al., ACL 2013) | 85.0 |
| CVG (SU-RNN) (Socher et al., ACL 2013) | 90.4 |
| Charniak - Self Trained (McClosky et al. 2006) | 91.0 |
| Charniak - Self Trained-ReRanked (McClosky et al. 2006) | 92.1 |

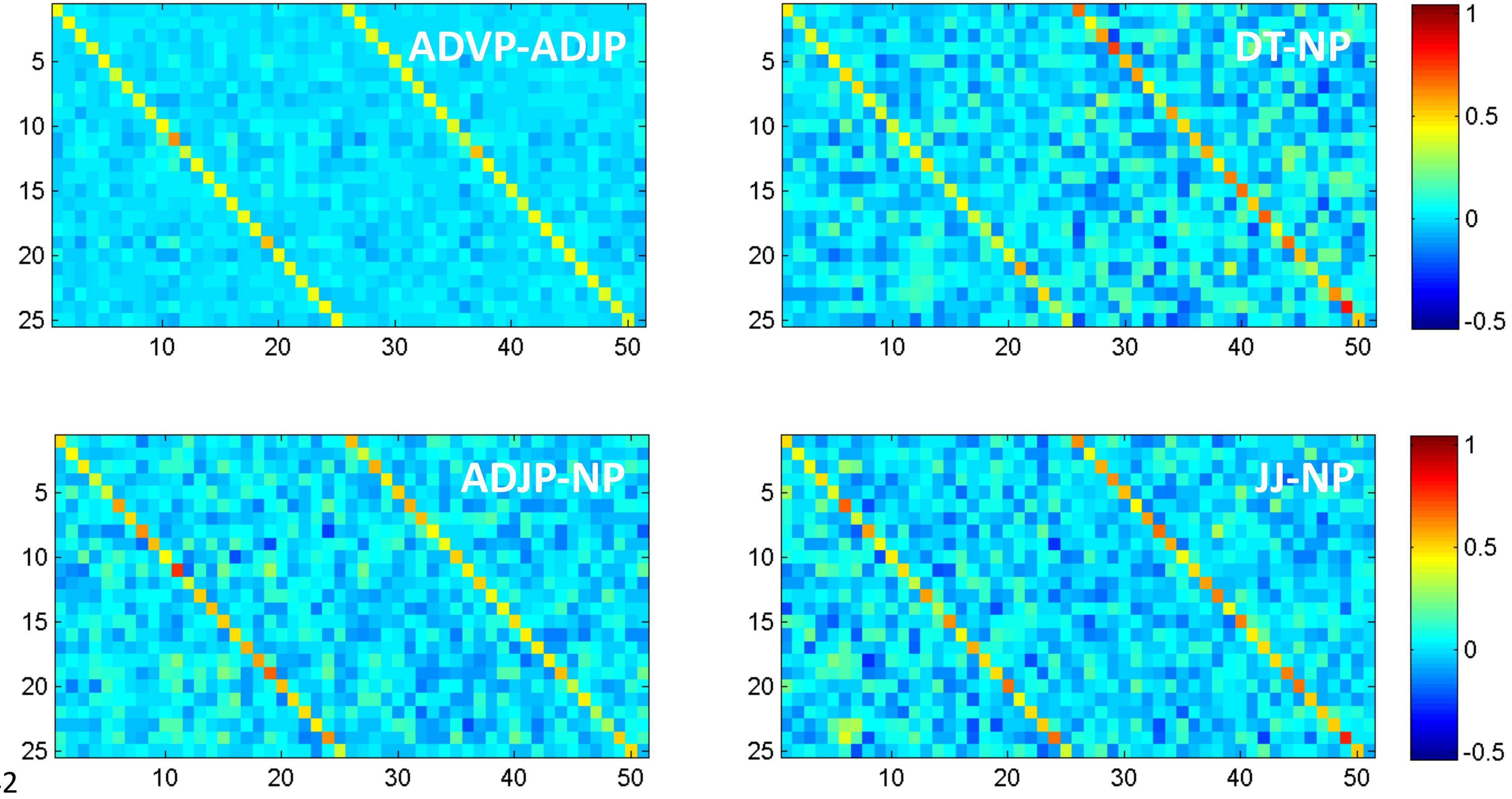
# SU-RNN / CVG [Socher, Bauer, Manning, Ng 2013]

## Learns soft notion of head words

Initialization: $W^{(\cdot\cdot)} = 0.5[I_{n\times n} I_{n\times n} 0_{n\times 1}] + \epsilon$



# SU-RNN / CVG [Socher, Bauer, Manning, Ng 2013]

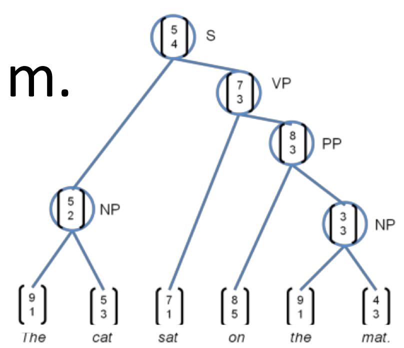


# Analysis of resulting vector representations

All the figures are adjusted for seasonal variations
1. All the numbers are adjusted for seasonal fluctuations
2. All the figures are adjusted to remove usual seasonal patterns

Knight-Ridder wouldn't comment on the offer
1. Harsco declined to say what country placed the order
2. Coastal wouldn't disclose the terms

Sales grew almost 7% to $UNK m. from $UNK m.
1. Sales rose more than 7% to $94.9 m. from $88.3 m.
2. Sales surged 40% to UNK b. yen from UNK b.



# Version 3:
# Compositionality Through Recursive Matrix-Vector Spaces

[Socher, Huval, Bhat, Manning, & Ng, 2012]

Before: $p = \tanh\left(W\begin{bmatrix}c_1\\c_2\end{bmatrix} + b\right)$

One way to make the composition function more powerful was by untying the weights *W*

But what if words act mostly as an operator, e.g. "very" in

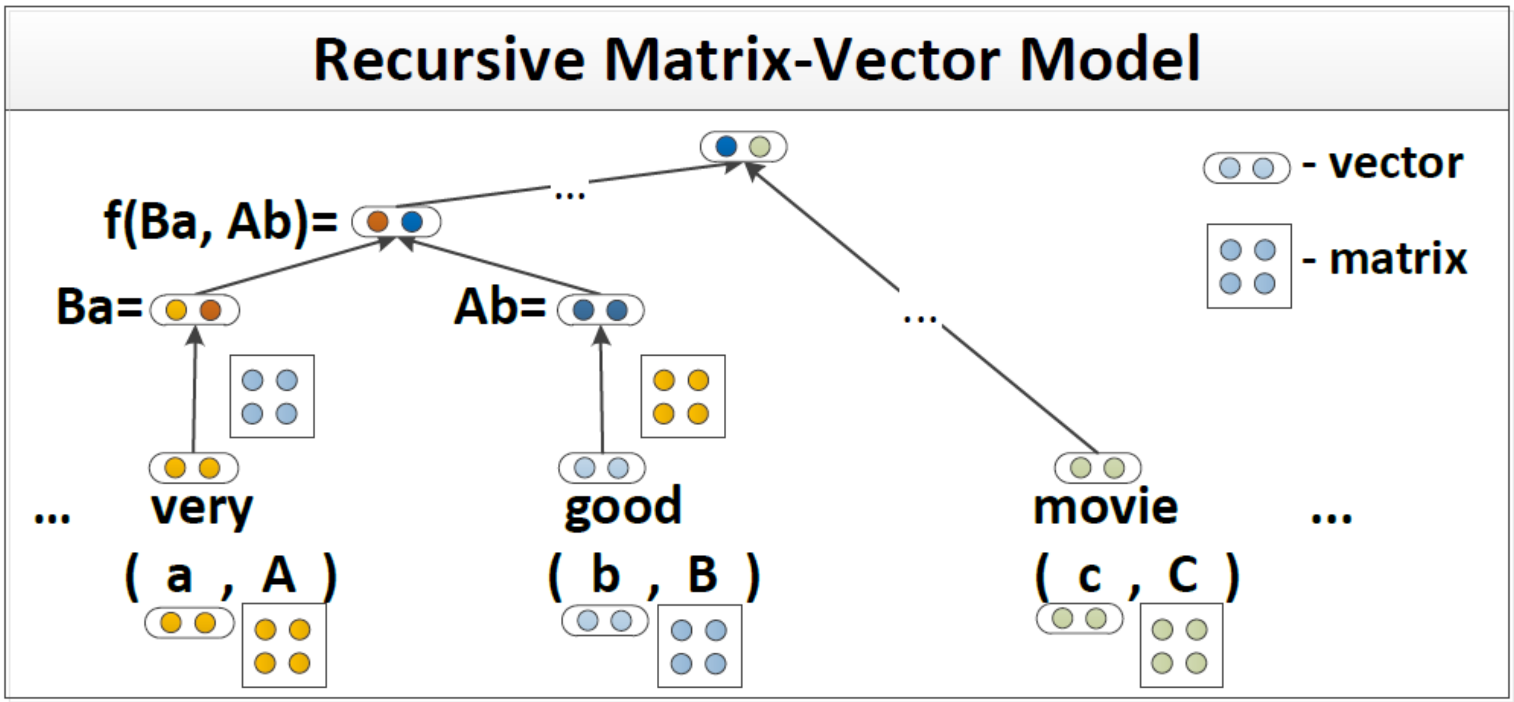
*very good*

Proposal: A new composition function

# Compositionality Through Recursive Matrix-Vector Recursive Neural Networks

$$p = \tanh\left(W\begin{bmatrix} c_1 \\ c_2 \end{bmatrix} + b\right)$$

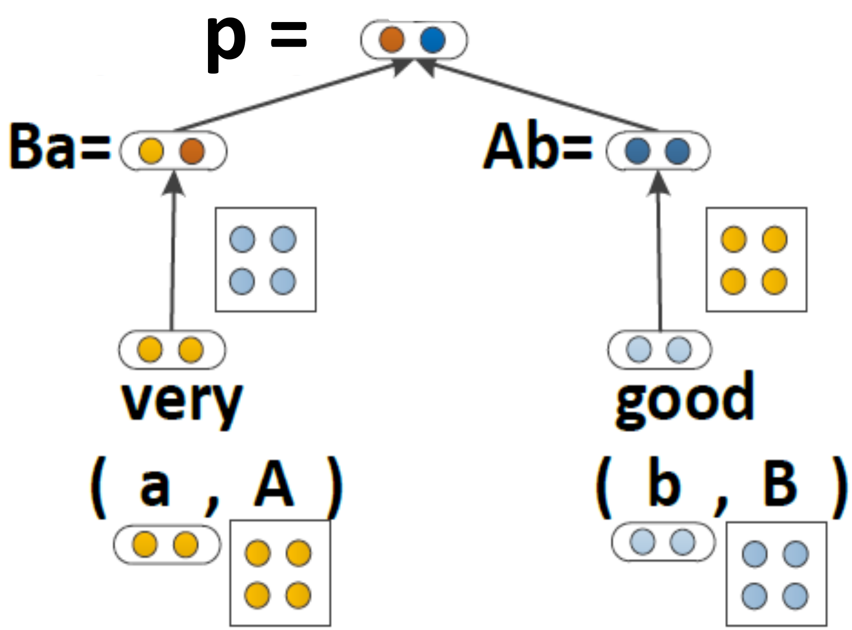
$$\boxed{p = \tanh\left(W\begin{pmatrix} C_2 c_1 \\ C_1 c_2 \end{pmatrix} + b\right)}$$



# Matrix-vector RNNs

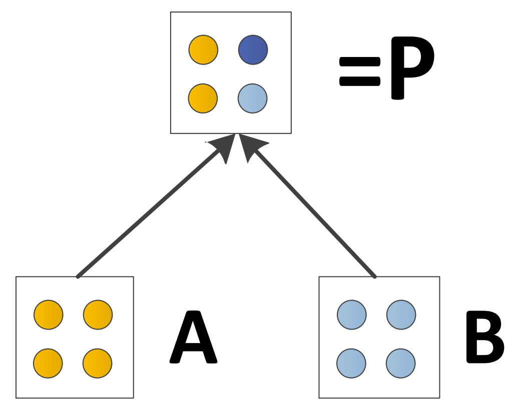
## [Socher, Huval, Bhat, Manning, & Ng, 2012]

$$p = f\left(W \begin{bmatrix} Ba \\ Ab \end{bmatrix}\right)$$



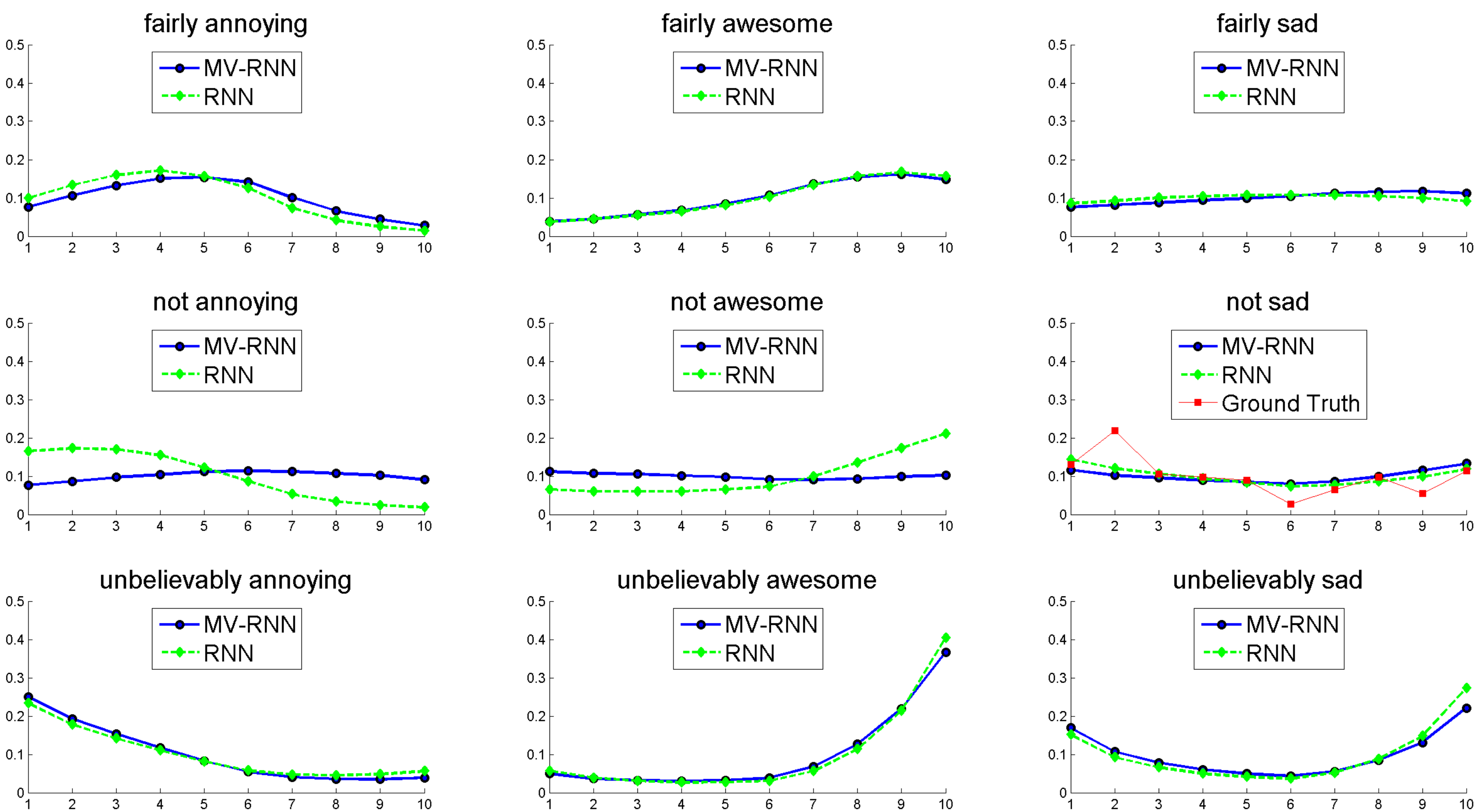

$$\boxed{P = g(A, B) = W_M \begin{bmatrix} A \\ B \end{bmatrix}}$$

$$W_M \in \mathbb{R}^{n \times 2n}$$



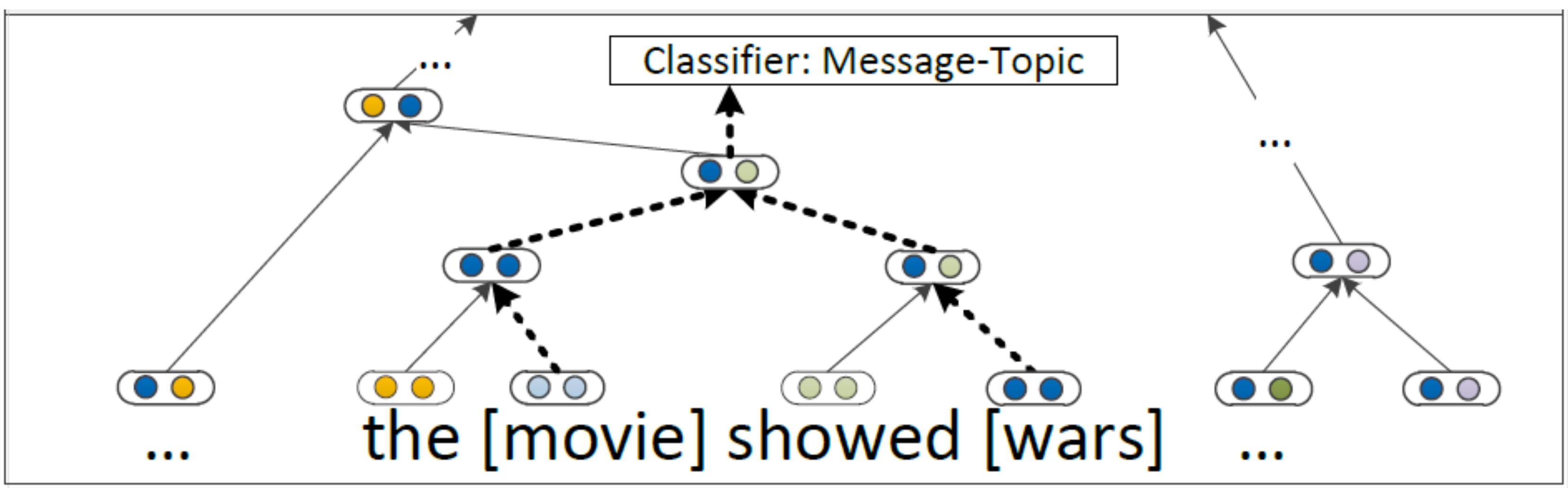
# Predicting Sentiment Distributions

Good example for non-linearity in language



# Classification of Semantic Relationships

- Can an MV-RNN learn how a large syntactic context conveys a semantic relationship?
- My [apartment]$_{e1}$ has a pretty large [kitchen] $_{e2}$
  → component-whole relationship (e2,e1)
- Build a single compositional semantics for the minimal constituent including both terms



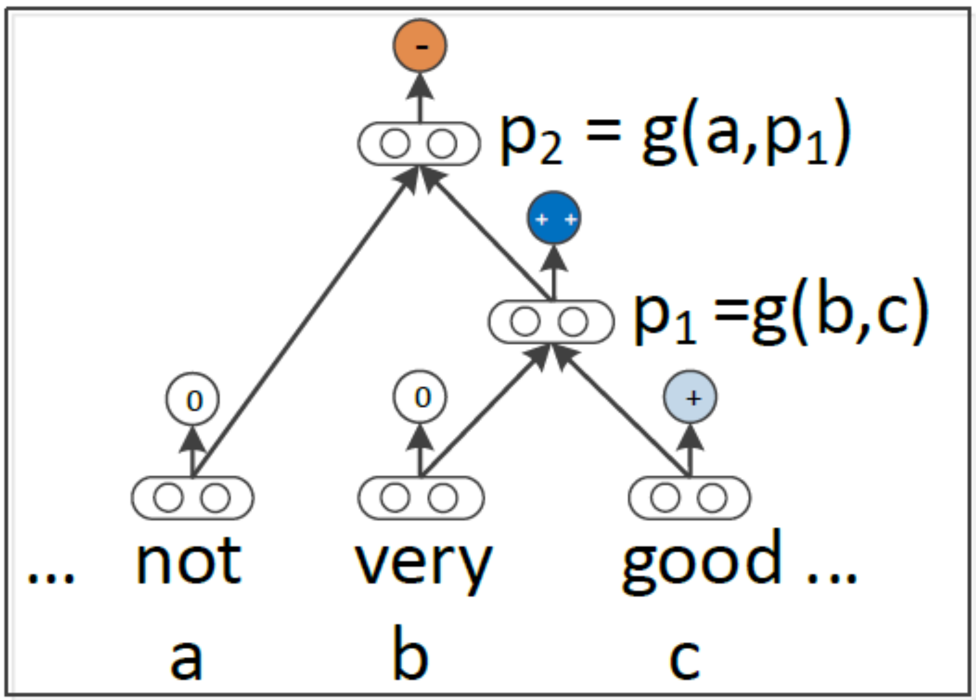
# Classification of Semantic Relationships

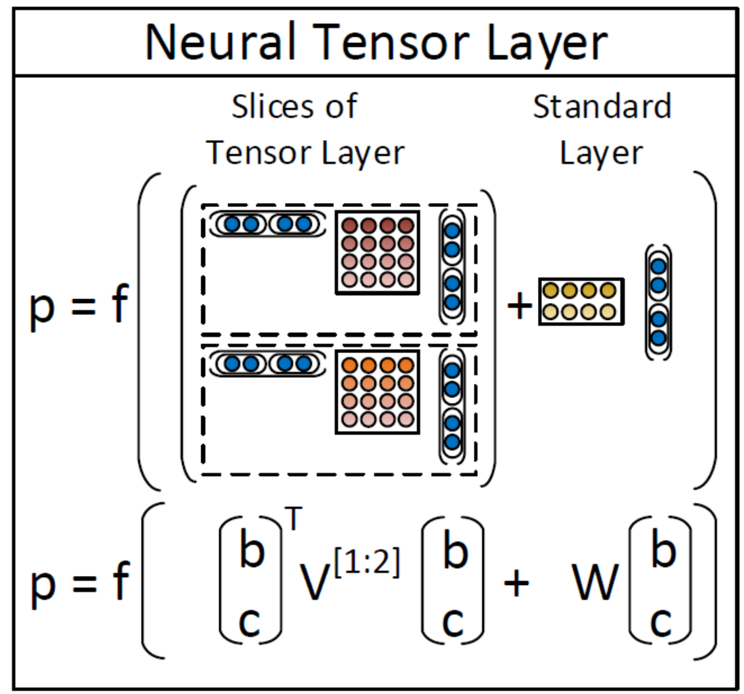
| Classifier | Features | F1 |
|---|---|---|
| SVM | POS, stemming, syntactic patterns | 60.1 |
| MaxEnt | POS, WordNet, morphological features, noun compound system, thesauri, Google n-grams | 77.6 |
| SVM | POS, WordNet, prefixes, morphological features, dependency parse features, Levin classes, PropBank, FrameNet, NomLex-Plus, Google n-grams, paraphrases, TextRunner | 82.2 |
| RNN | – | 74.8 |
| MV-RNN | – | 79.1 |
| **MV-RNN** | **POS, WordNet, NER** | **82.4** |

# Version 4: Recursive Neural Tensor Network

Socher, Perelygin, Wu, Chuang, Manning, Ng, and Potts 2013

- Less parameters than MV-RNN
- Allows the two word or phrase vectors to interact multiplicatively



# Beyond the bag of words: Sentiment detection

Is the tone of a piece of text positive, negative, or neutral?

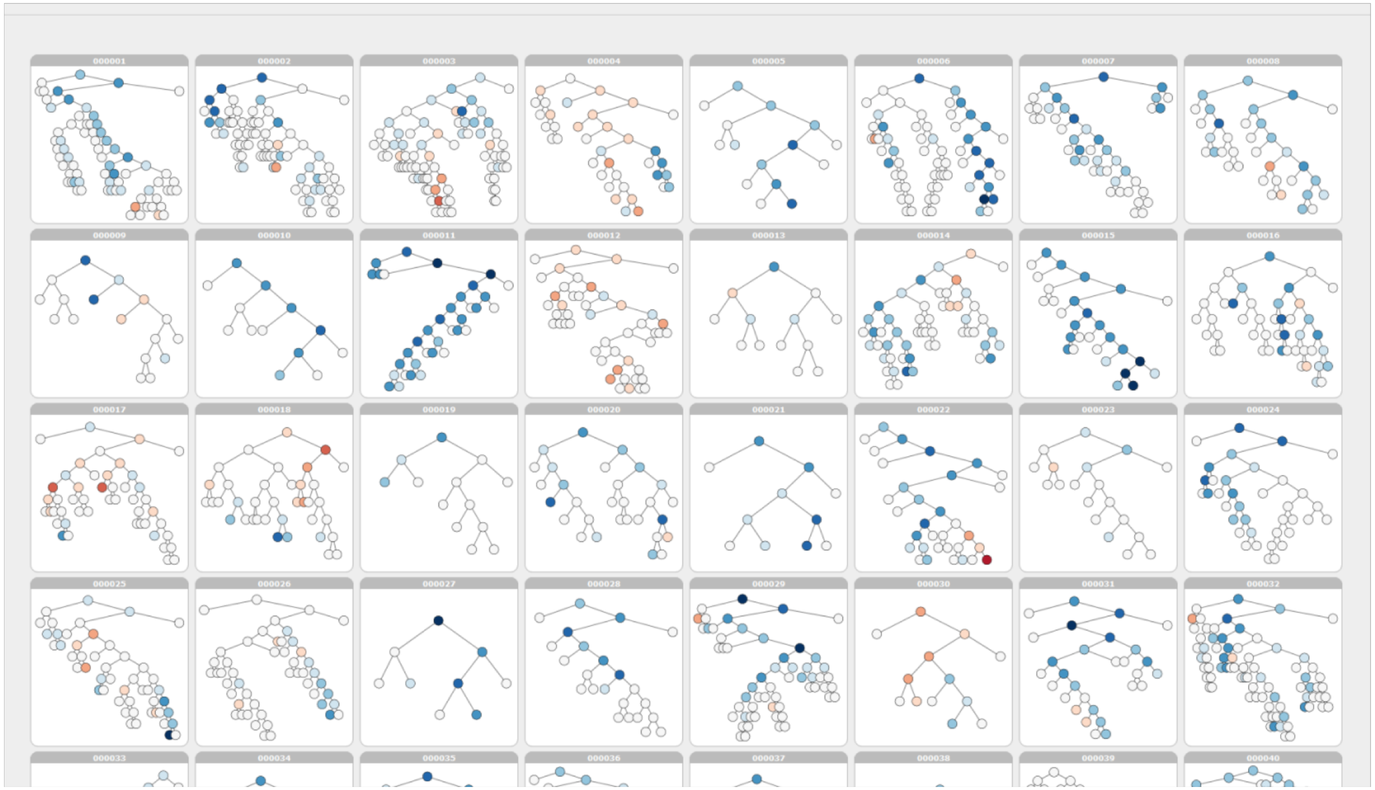
- Sentiment is that sentiment is “easy”
- Detection accuracy for longer documents ~90%, BUT

… … loved … … … … … great … … … … … … impressed … … … … … … marvelous … … … …

With this cast, and this subject matter, the movie should have been funnier and more entertaining.

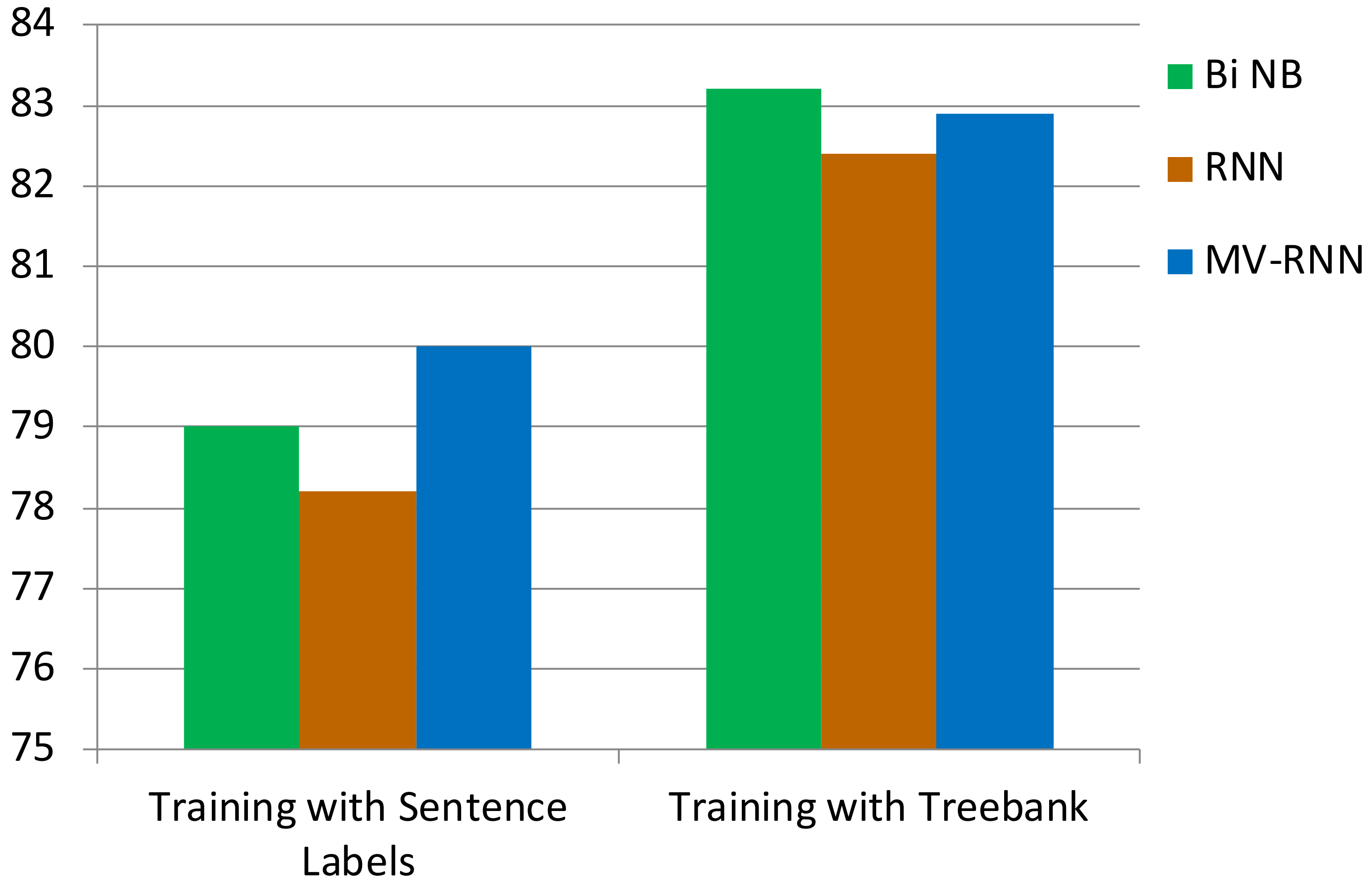
# Stanford Sentiment Treebank

- 215,154 phrases labeled in 11,855 sentences
- Can actually train and test compositions



http://nlp.stanford.edu:8080/sentiment/

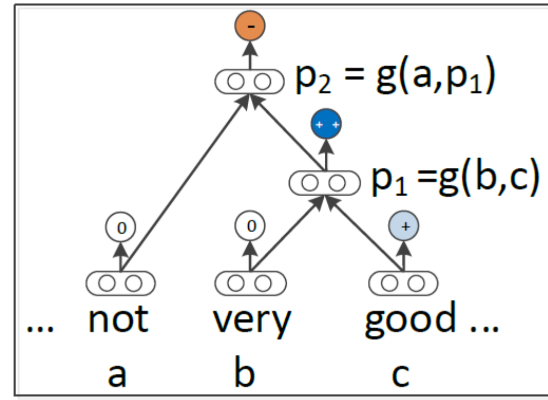
# Better Dataset Helped All Models

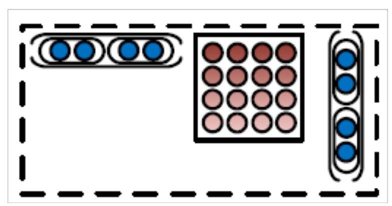



- Hard negation cases are still mostly incorrect
- We also need a more powerful model!

# Version 4: Recursive Neural Tensor Network

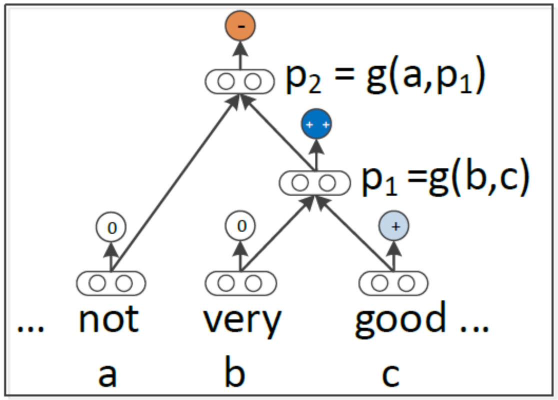
Idea: Allow both additive and mediated multiplicative interactions of vectors

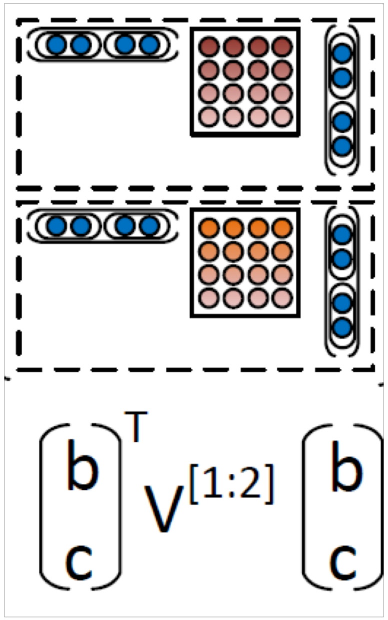


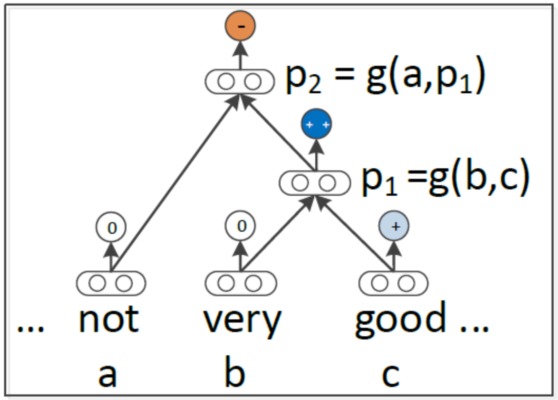
$$\begin{bmatrix} b \\ c \end{bmatrix}^T V \begin{bmatrix} b \\ c \end{bmatrix}$$

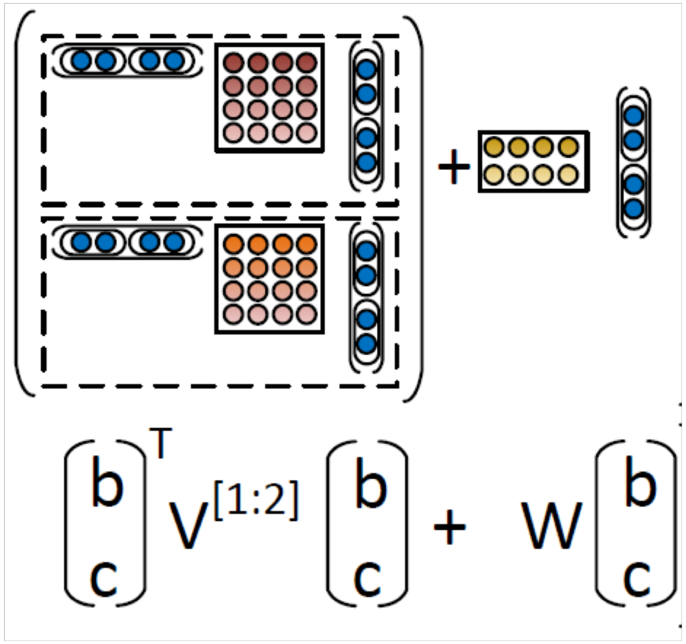
# Recursive Neural Tensor Network



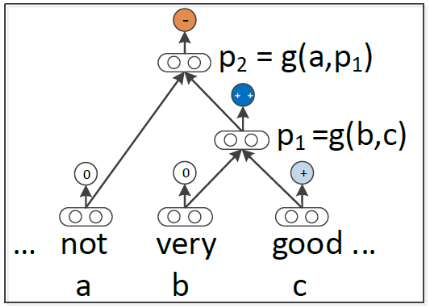
# Recursive Neural Tensor Network

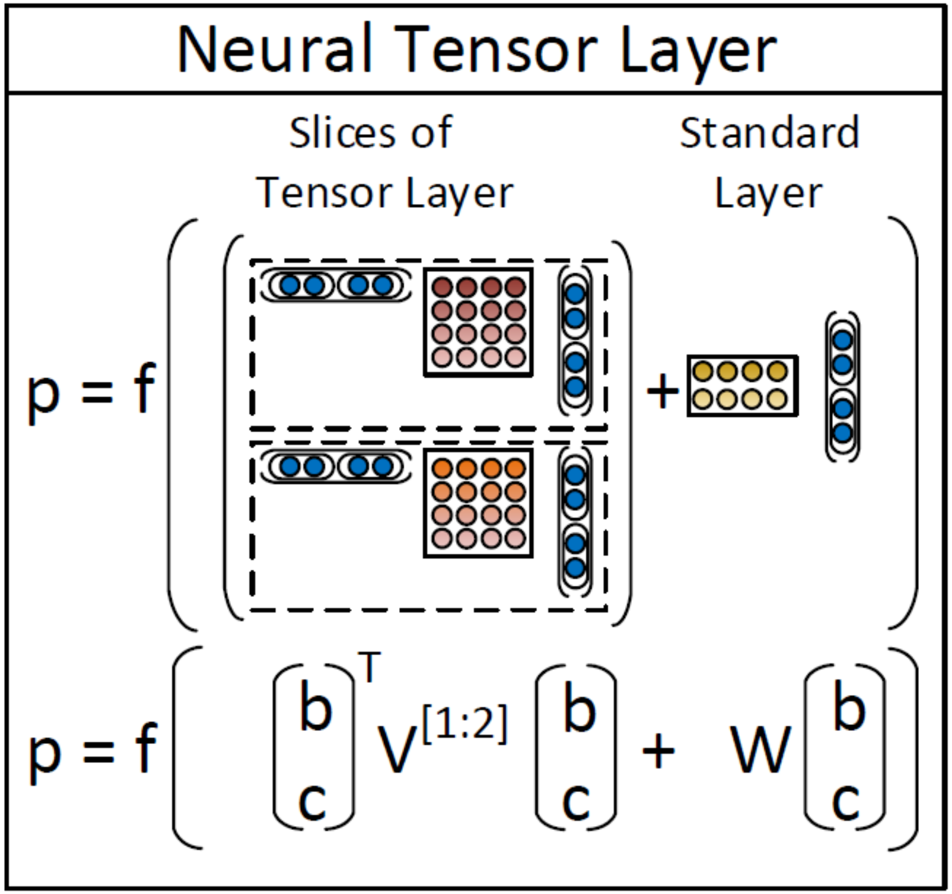


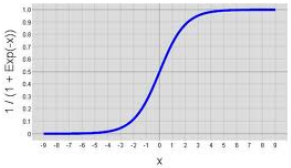



# Recursive Neural Tensor Network

- Use resulting vectors in tree as input to a classifier like logistic regression
- Train all weights jointly with gradient descent

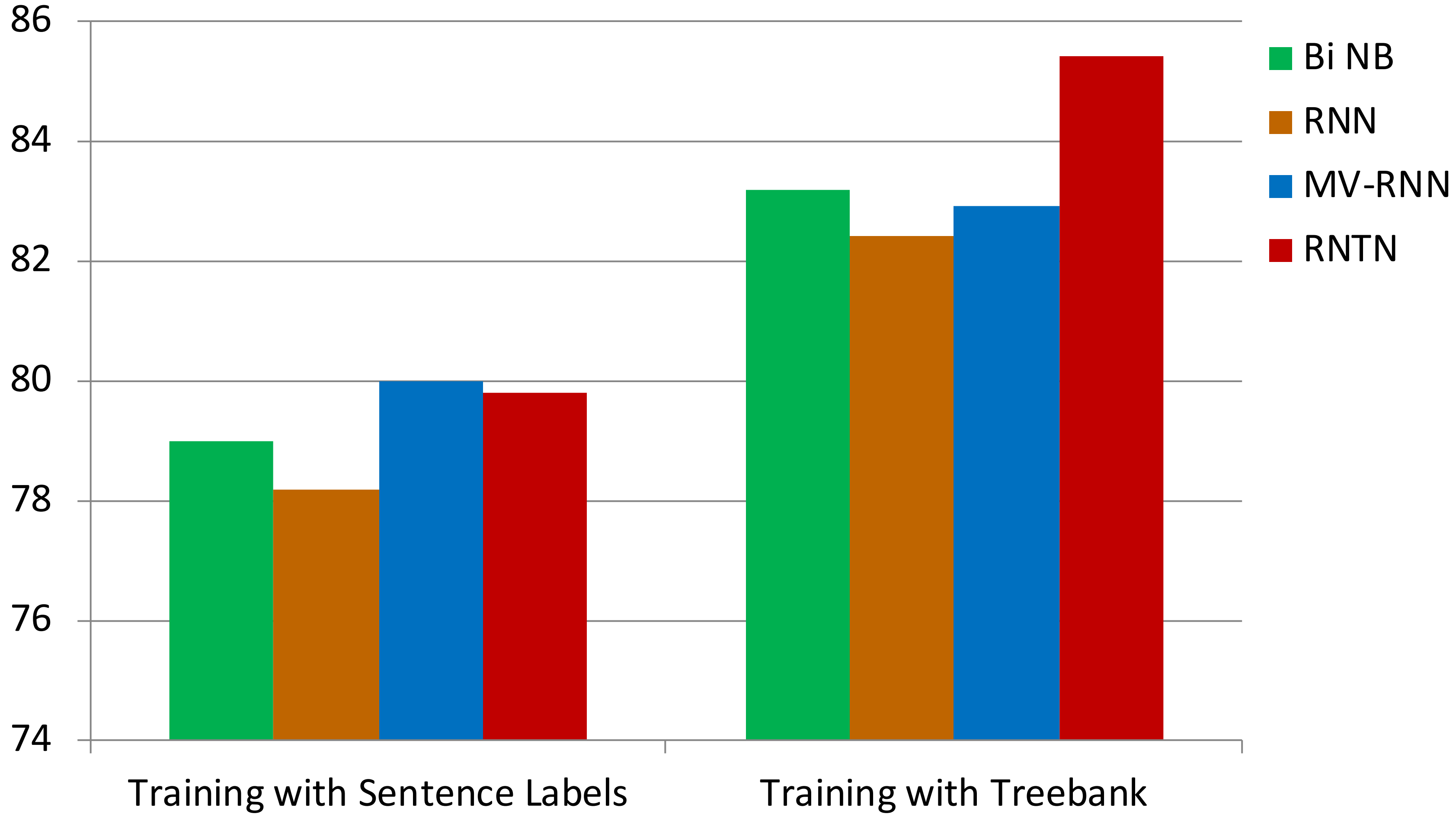


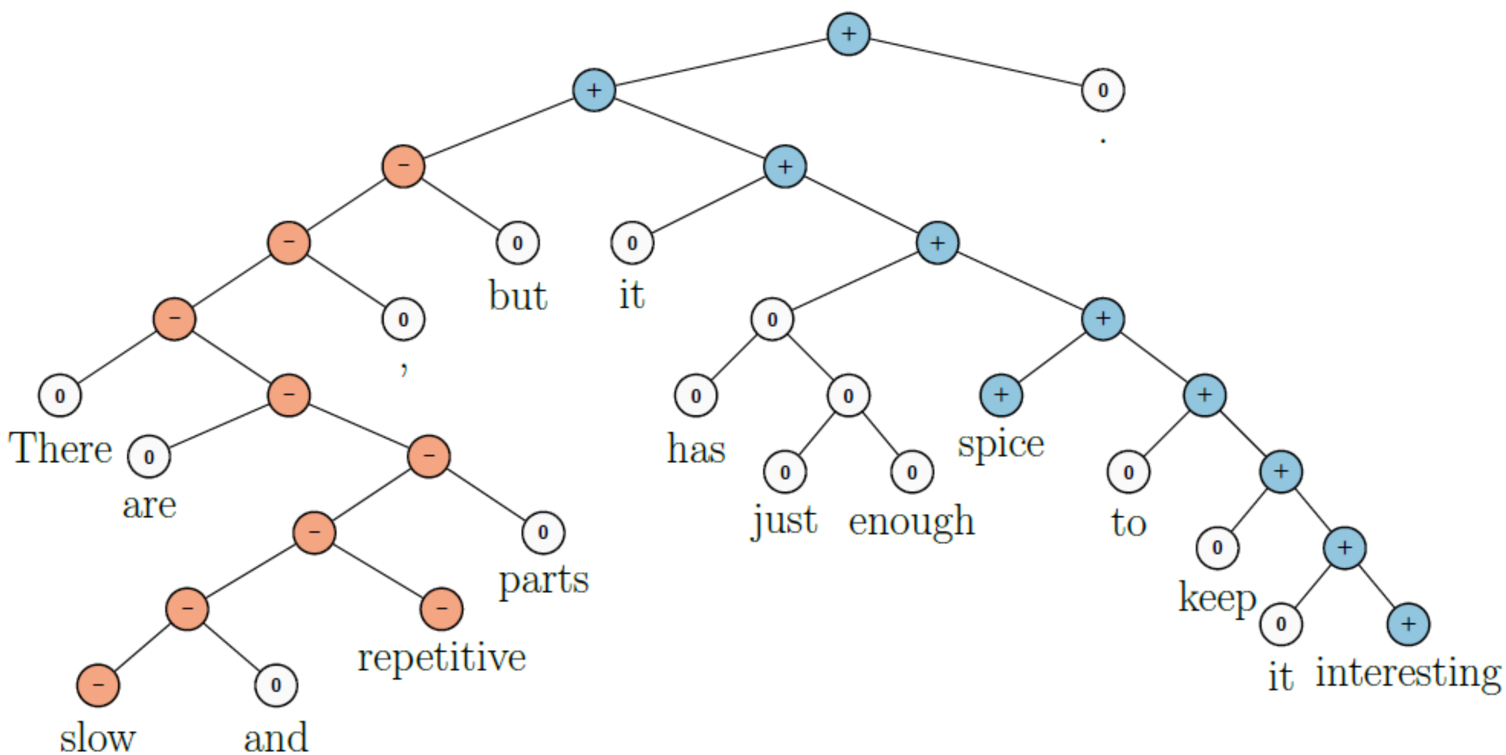
# Positive/Negative Results on Treebank

Classifying Sentences: Accuracy improves to 85.4



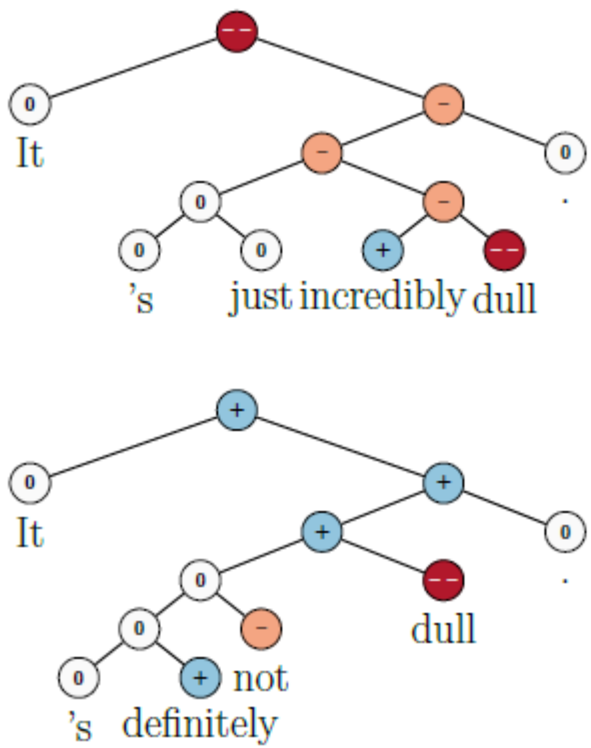
# Experimental Results on Treebank

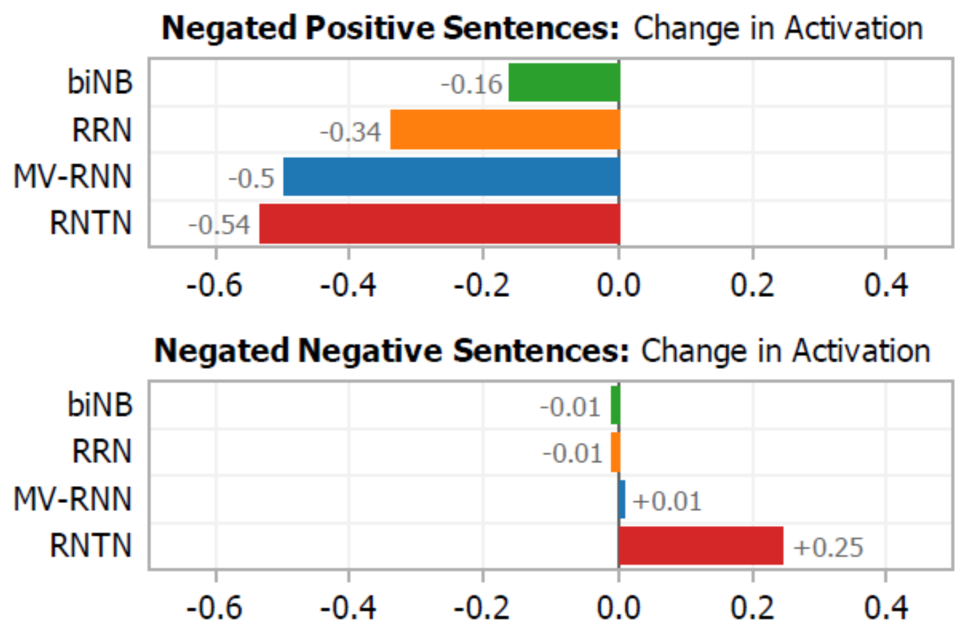
- RNTN can capture constructions like *X but Y*
- RNTN accuracy of 72%, compared to MV-RNN (65%), biword NB (58%) and RNN (54%)



# Negation Results

When negating negatives, positive activation should increase!




**Demo: http://nlp.stanford.edu:8080/sentiment/**

# Version 5:
# Improving Deep Learning Semantic Representations using a TreeLSTM

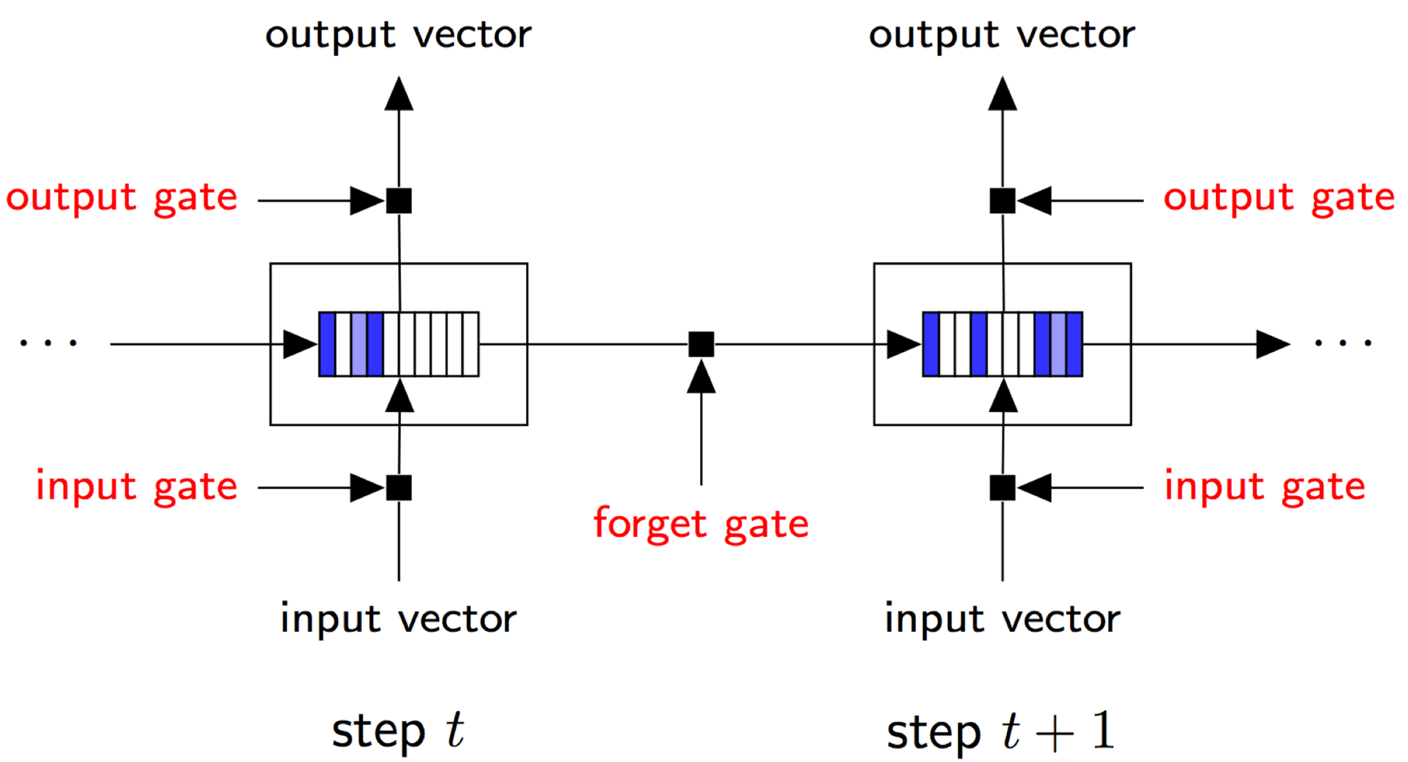
**[Tai et al., ACL 2015; also Zhu et al. ICML 2015]**

Goals:

- Still trying to represent the meaning of a sentence as a location in a (high-dimensional, continuous) vector space
- In a way that accurately handles semantic composition and sentence meaning
- Generalizing the widely used chain-structured LSTM to trees

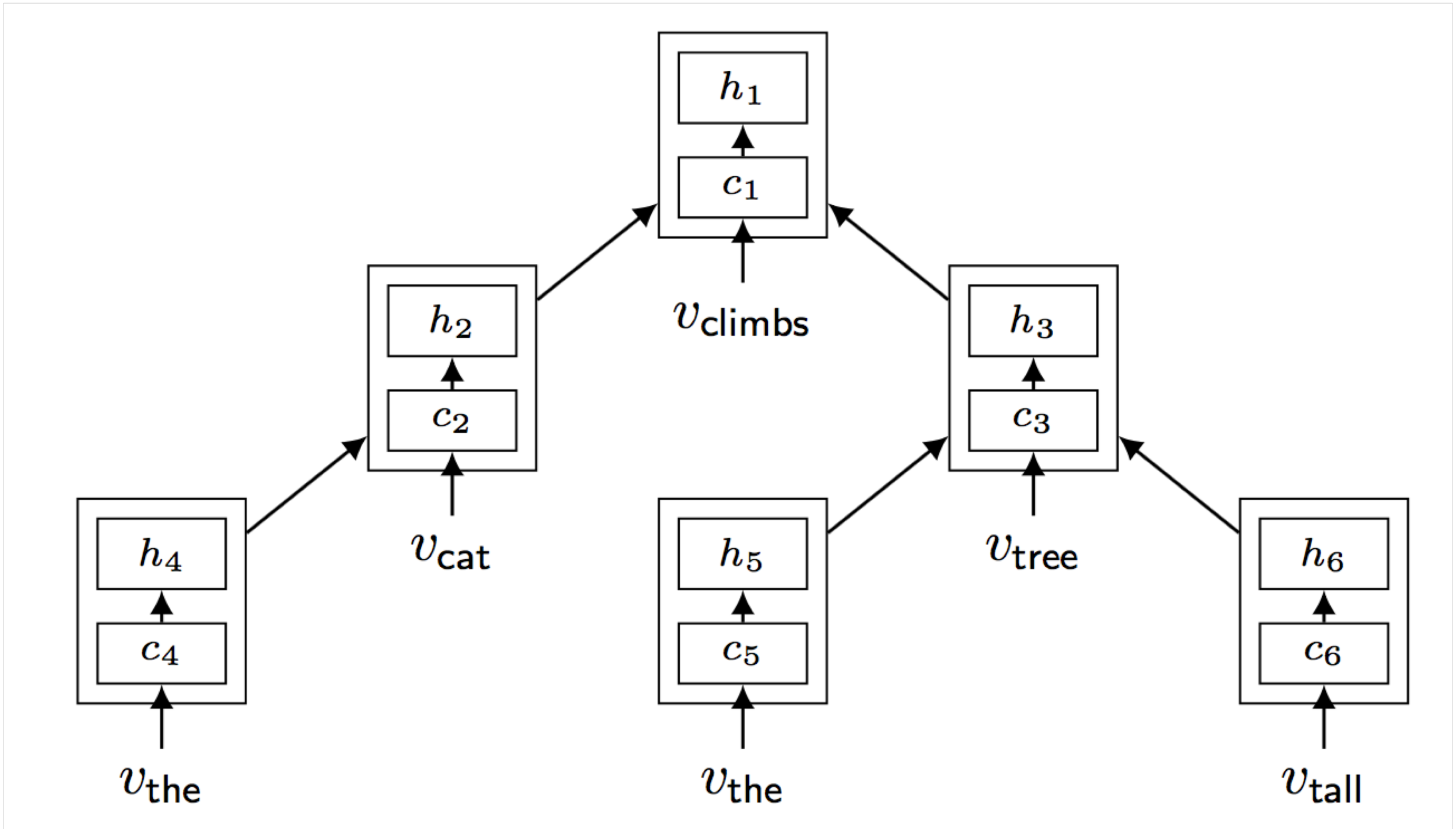
# Long Short-Term Memory (LSTM) Units for Sequential Composition

Gates are vectors in $[0,1]^d$ multiplied element-wise for soft masking



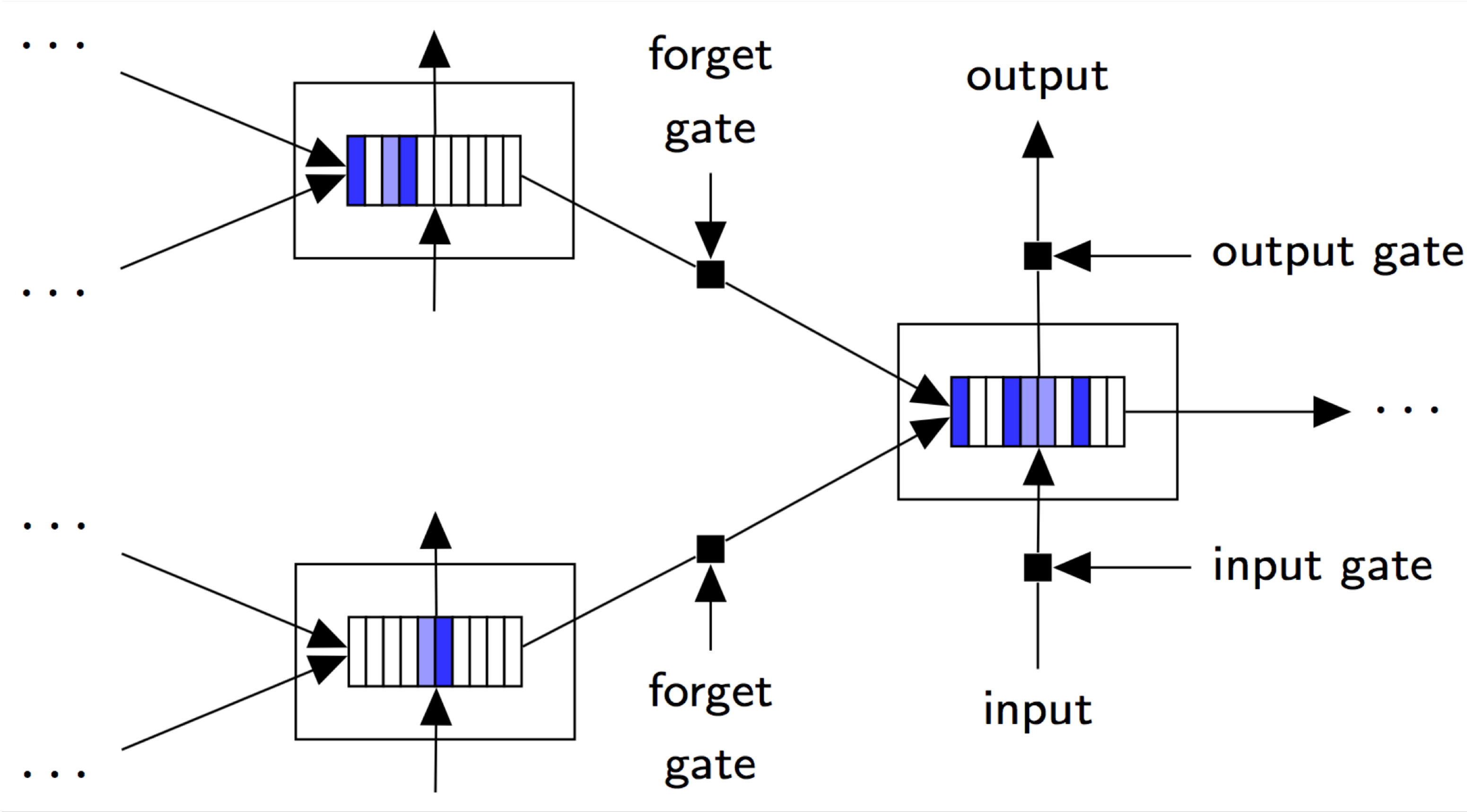
# Tree-Structured Long Short-Term Memory Networks

[Tai et al., ACL 2015]



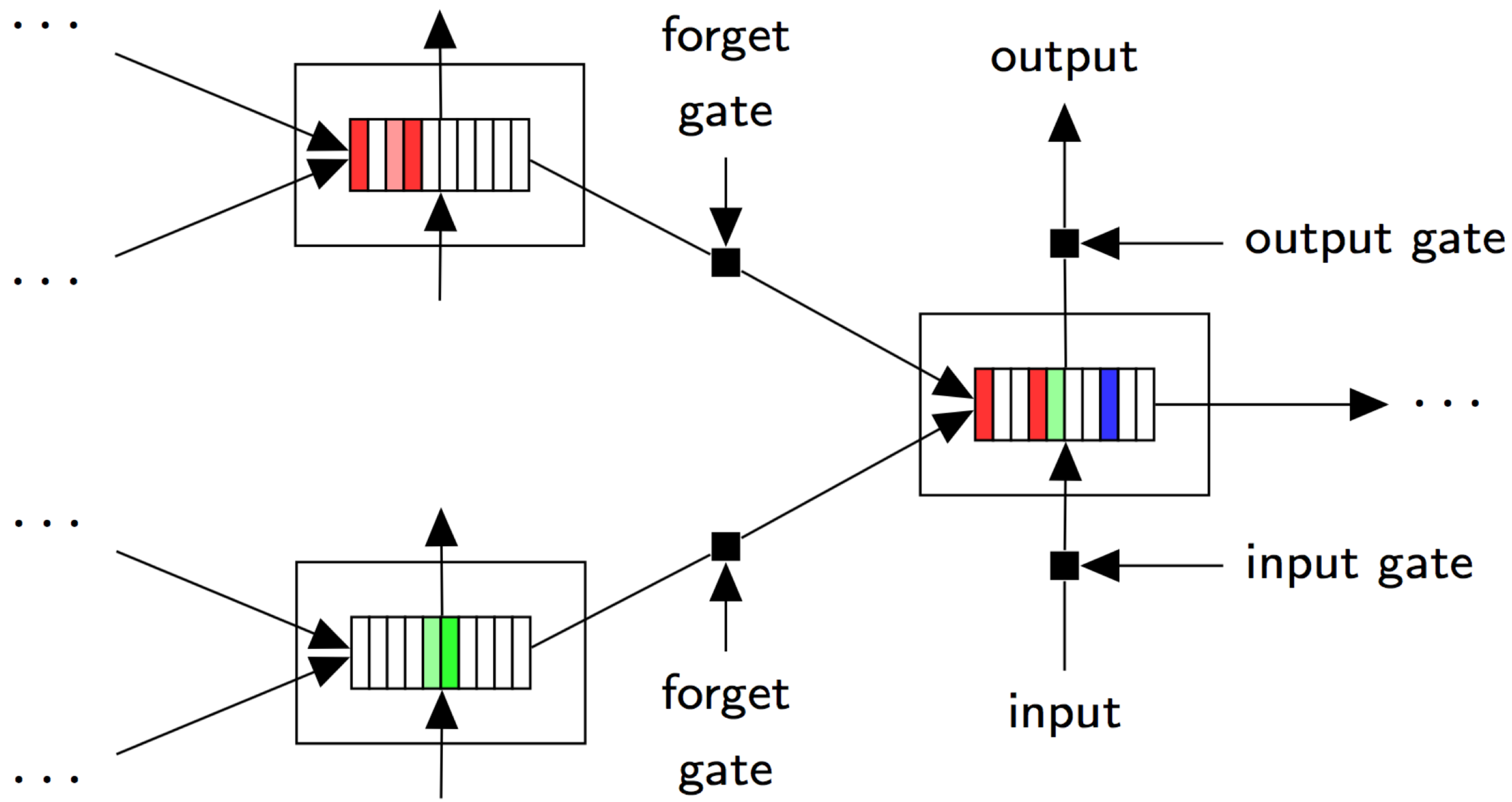
# Tree-structured LSTM

Generalizes sequential LSTM to trees with any branching factor



# Tree-structured LSTM

Generalizes sequential LSTM to trees with any branching factor



# Results: Sentiment Analysis: Stanford Sentiment Treebank

| Method | Accuracy % (Fine-grain, 5 classes) |
|---|---|
| RNTN (Socher et al. 2013) | 45.7 |
| Paragraph-Vec (Le & Mikolov 2014) | 48.7 |
| DRNN (Irsoy & Cardie 2014) | 49.8 |
| LSTM | 46.4 |
| Tree LSTM (this work) | **50.9** |

# Results: Sentiment Analysis: Stanford Sentiment Treebank

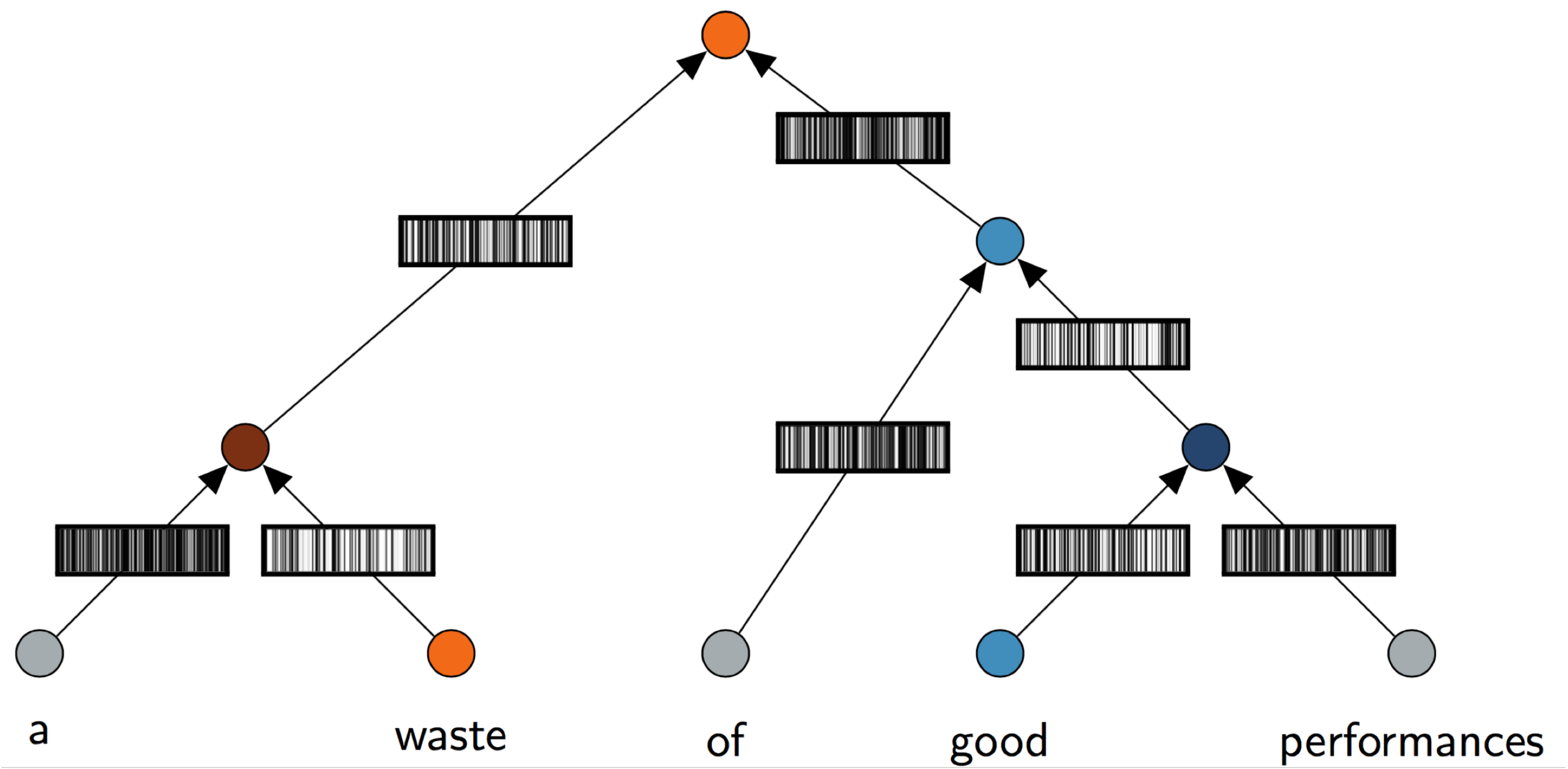
| Method | Accuracy % (Pos/Neg) |
|---|---|
| RNTN (Socher et al. 2013) | 85.4 |
| Paragraph-Vec (Le & Mikolov 2014) | 87.8 |
| DRNN (Irsoy & Cardie 2014) | 86.6 |
| LSTM | 84.9 |
| Tree LSTM (this work) | **88.0** |

# Results: Semantic Relatedness
## SICK 2014 (Sentences Involving Compositional Knowledge)

| Method | Pearson correlation |
|---|---|
| Word vector average | 0.758 |
| Meaning Factory (Bjerva et al. 2014) | 0.827 |
| ECNU (Zhao et al. 2014) | 0.841 |
| LSTM | 0.853 |
| Tree LSTM | **0.868** |

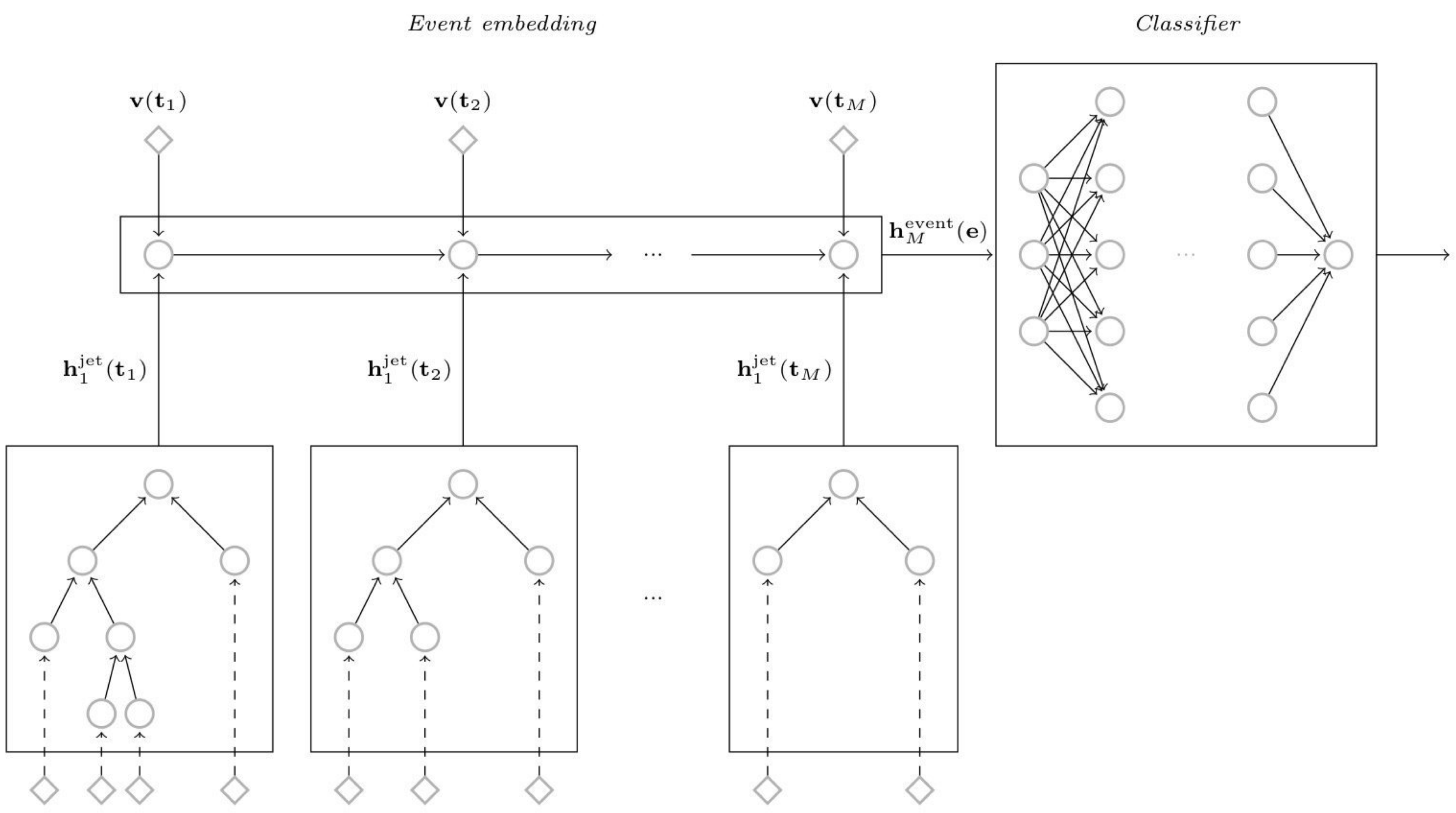
# Forget Gates: Selective State Preservation

- Stripes = forget gate activations; more white ⇒ more preserved



# 5. QCD-Aware Recursive Neural Networks for Jet Physics

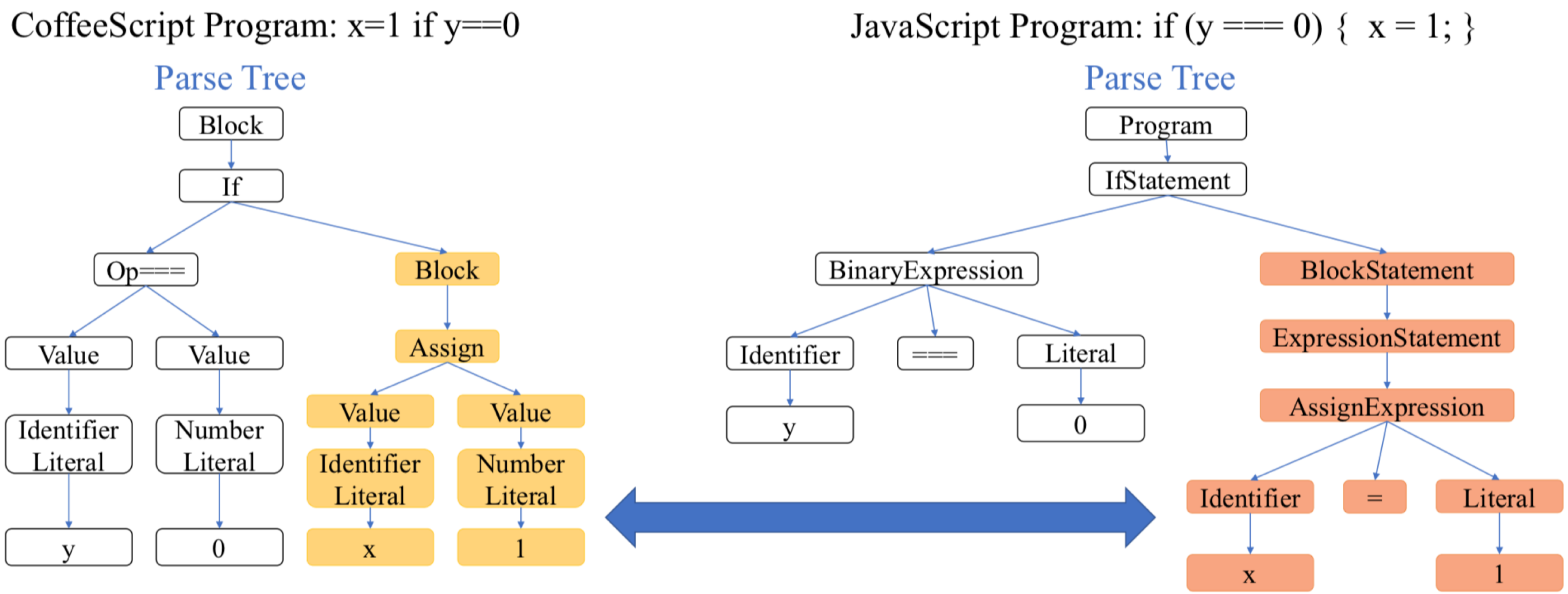
## Gilles Louppe, Kyunghun Cho, Cyril Becot, Kyle Cranmer (2017)



# Tree-to-tree Neural Networks for Program Translation

## [Chen, Liu, and Song NeurIPS 2018]

- Explores using tree-structured encoding **and generation** for translation between programming languages
- In generation, you use attention over the source tree



# Tree-to-tree Neural Networks for Program Translation

## [Chen, Liu, and Song NeurIPS 2018]

| | Tree2tree | | | Seq2seq | | | | Seq2tree | | Tree2seq | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | T→T | T→T (-PF) | T→T (-Attn) | P→P | P→T | T→P | T→T | P→T | T→T | T→P | T→T |
| CoffeeScript to JavaScript translation | | | | | | | | | | | |
| CJ-AS | **99.57%** | 98.80% | 0.09% | 90.51% | 79.82% | 92.73% | 89.13% | 86.52% | 88.50% | 96.96% | 92.18% |
| CJ-BS | **99.75%** | 99.67% | 0% | 97.44% | 16.26% | 98.05% | 93.89% | 91.97% | 88.22% | 96.83% | 78.77% |
| CJ-AL | **97.15%** | 71.52% | 0% | 21.04% | 0% | 0% | 0% | 80.82% | 78.60% | 82.55% | 46.94% |
| CJ-BL | **95.60%** | 78.61% | 0% | 19.26% | 9.98% | 25.35% | 42.08% | 76.12% | 76.21% | 83.61% | 26.83% |
| JavaScript to CoffeeScript translation | | | | | | | | | | | |
| JC-AS | **87.75%** | 85.11% | 0.09% | 83.07% | 86.13% | 73.88% | 86.31% | 86.86% | 86.99% | 71.61% | 86.53% |
| JC-BS | **86.37%** | 80.35% | 0% | 80.49% | 85.94% | 69.77% | 85.28% | 85.06% | 84.25% | 66.82% | 85.31% |
| JC-AL | **78.59%** | 54.93% | 0% | 77.10% | 77.30% | 65.52% | 75.70% | 77.11% | 77.59% | 60.75% | 75.75% |
| JC-BL | **75.62%** | 44.40% | 0% | 73.14% | 73.96% | 61.92% | 74.51% | 74.34% | 71.56% | 57.09% | 73.86% |

# Tree-to-tree Neural Networks for Program Translation

## [Chen, Liu, and Song NeurIPS 2018]

| | Tree2tree | J2C# | 1pSMT | mppSMT |
|---|---|---|---|---|
| | | Reported in [22] | | |
| Lucene | **72.8%** | 21.5% | 21.6% | 40.0% |
| POI | **72.2%** | 18.9% | 34.6% | 48.2% |
| Itext | **67.5%** | 25.1% | 24.4% | 40.6% |
| JGit | **68.7%** | 10.7% | 23.0% | 48.5% |
| JTS | **68.2%** | 11.7% | 18.5% | 26.3% |
| Antlr | 31.9% (**58.3%**) | 10.0% | 11.5% | 49.1% |

# Stanford Institute for Human-Centered Artificial Intelligence (HAI)

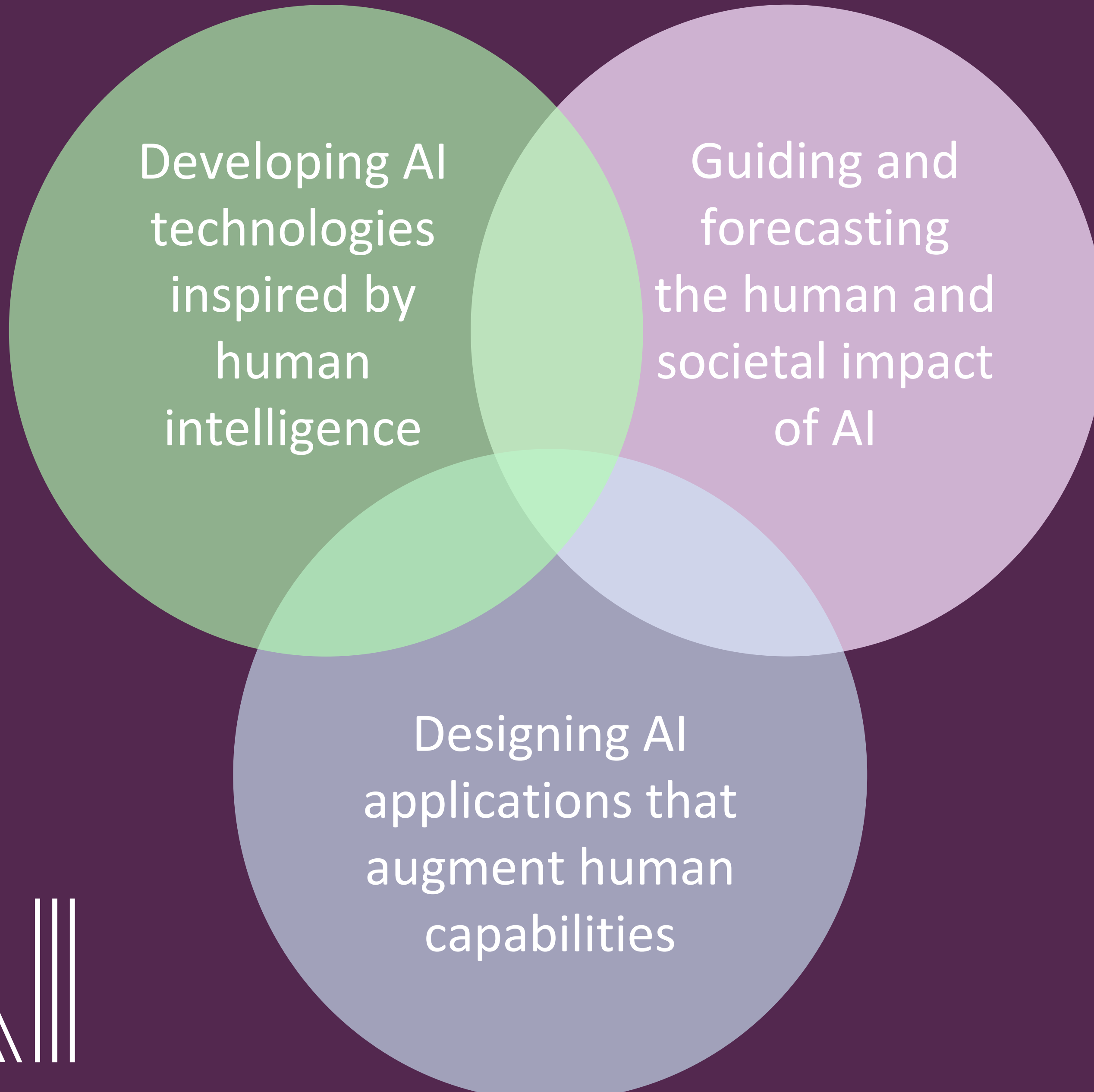
HAI

Stanford

# Human-Centered Artificial Intelligence

Artificial intelligence is poised to transform economies and societies, change the way we communicate and work, reshape governance and politics, and challenge the international order

HAI’s mission is to advance AI research, education, policy, and practice to improve the human condition




Stanford
Human-Centered
Artificial Intelligence
HAI