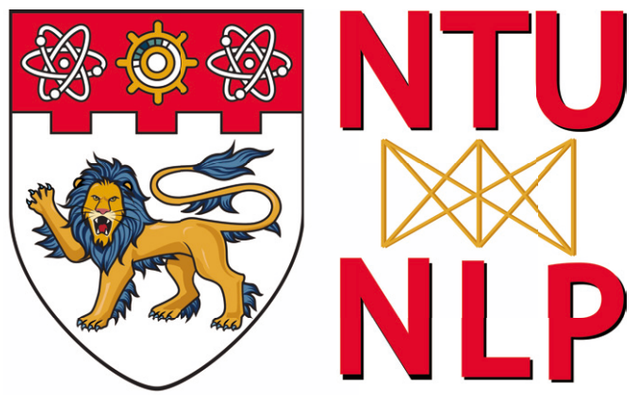
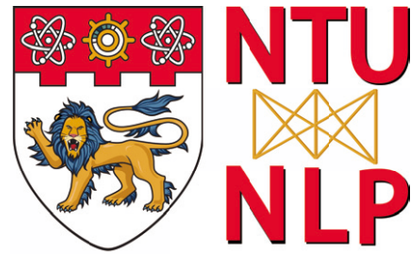
Deep Learning for Natural Language Processing

Shafiq Joty



Lecture 7: Recursive Neural Nets & Parsing

Where we are



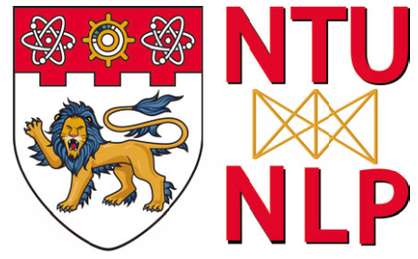
Models/Algorithms

- Linear models
- Feed-forward Neural Nets (FNN)
- Window-based methods
- Convolutional Nets
- Recurrent Neural Nets
- **Recursive Neural Nets**

NLP tasks/applications

- Word meaning
- Language modelling
- Sequence tagging
- Sequence encoding
- **Parsing**
- **Hierarchical encoding**

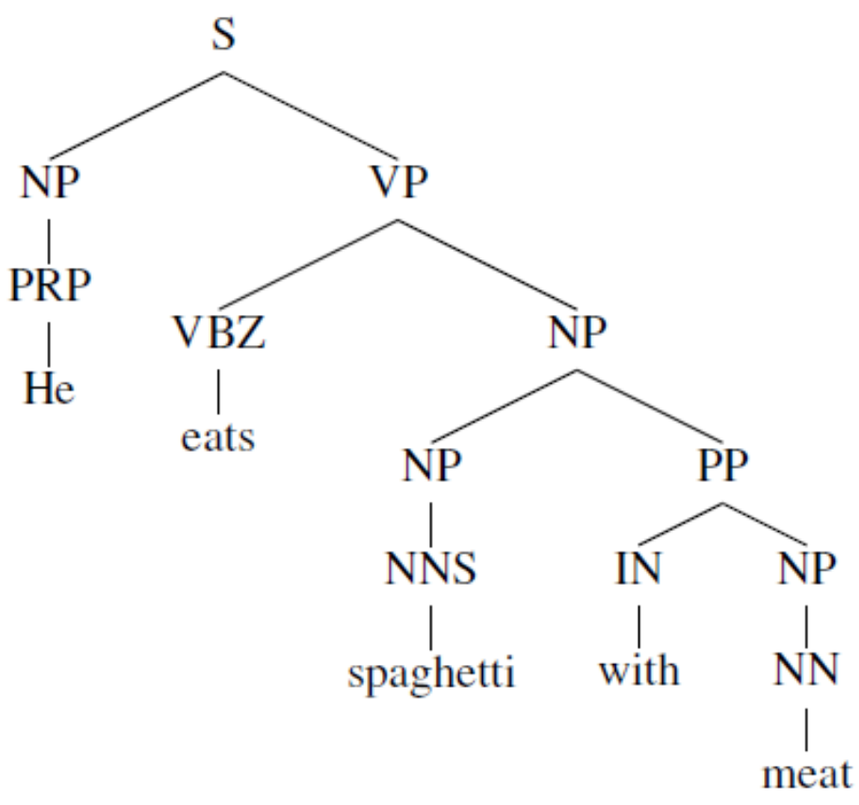
Today's Plan



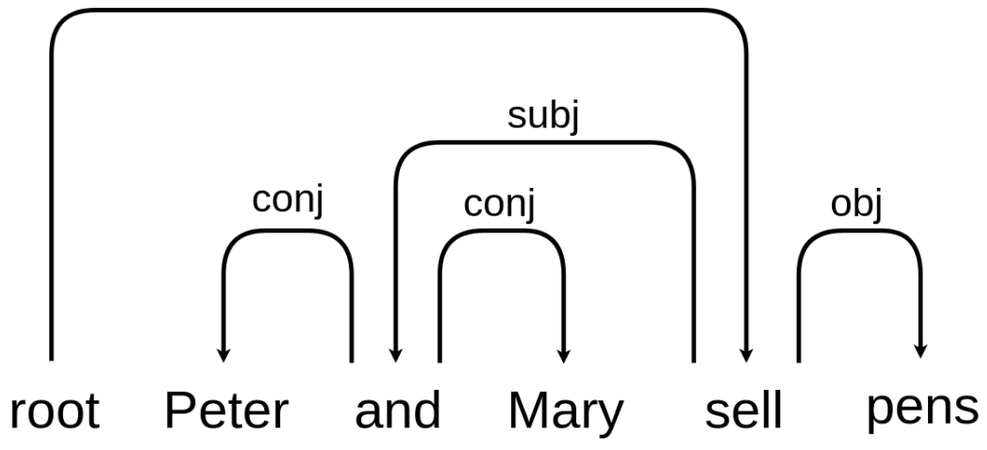
- Syntactic structures
  - Constituency & Dependency structures
- Need for syntactic structures
- Are languages hierarchical?
- Parsing with Recursive Neural Networks
- Backpropagation through tree
- Semantic composition with Tree RNNs
- Recent Parsers

Two Views of Syntactic Structures

Two types of syntactic structures of a sentence



Constituency/phrase structure

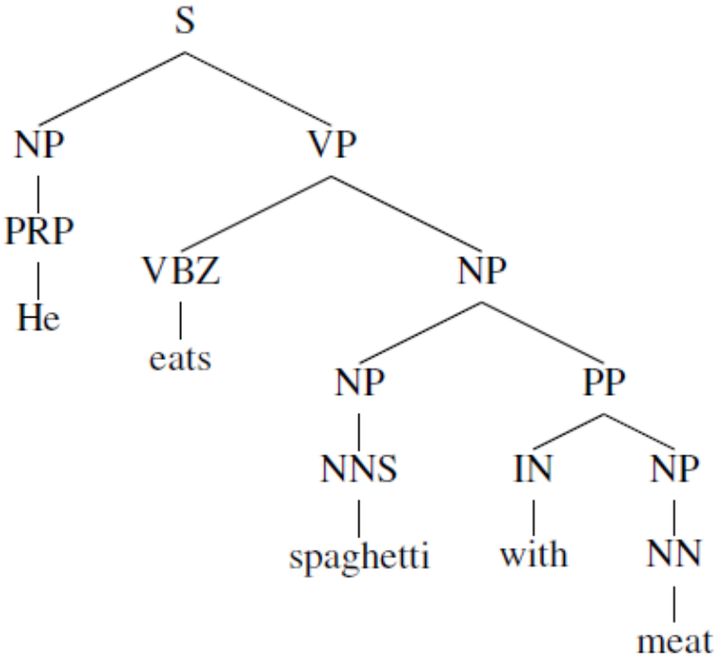


Dependency structure

Constituency Structure

- Phrase structure organizes words into nested constituents

- Atomic units are words. E.g., He, eats, spaghetti, with, meat
- Words combine into phrases: with meat,
- Phrases combine into larger phrases (recursively): spaghetti with meat

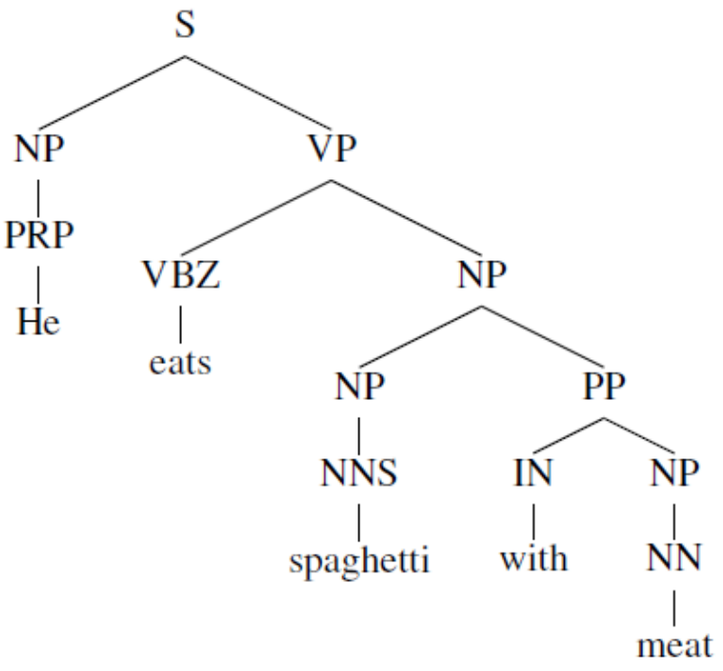


Source: Stanford 224n

Constituency Structure

- Phrase structure organizes words into nested constituents
  - Can represent the process (grammar) with CFG rules

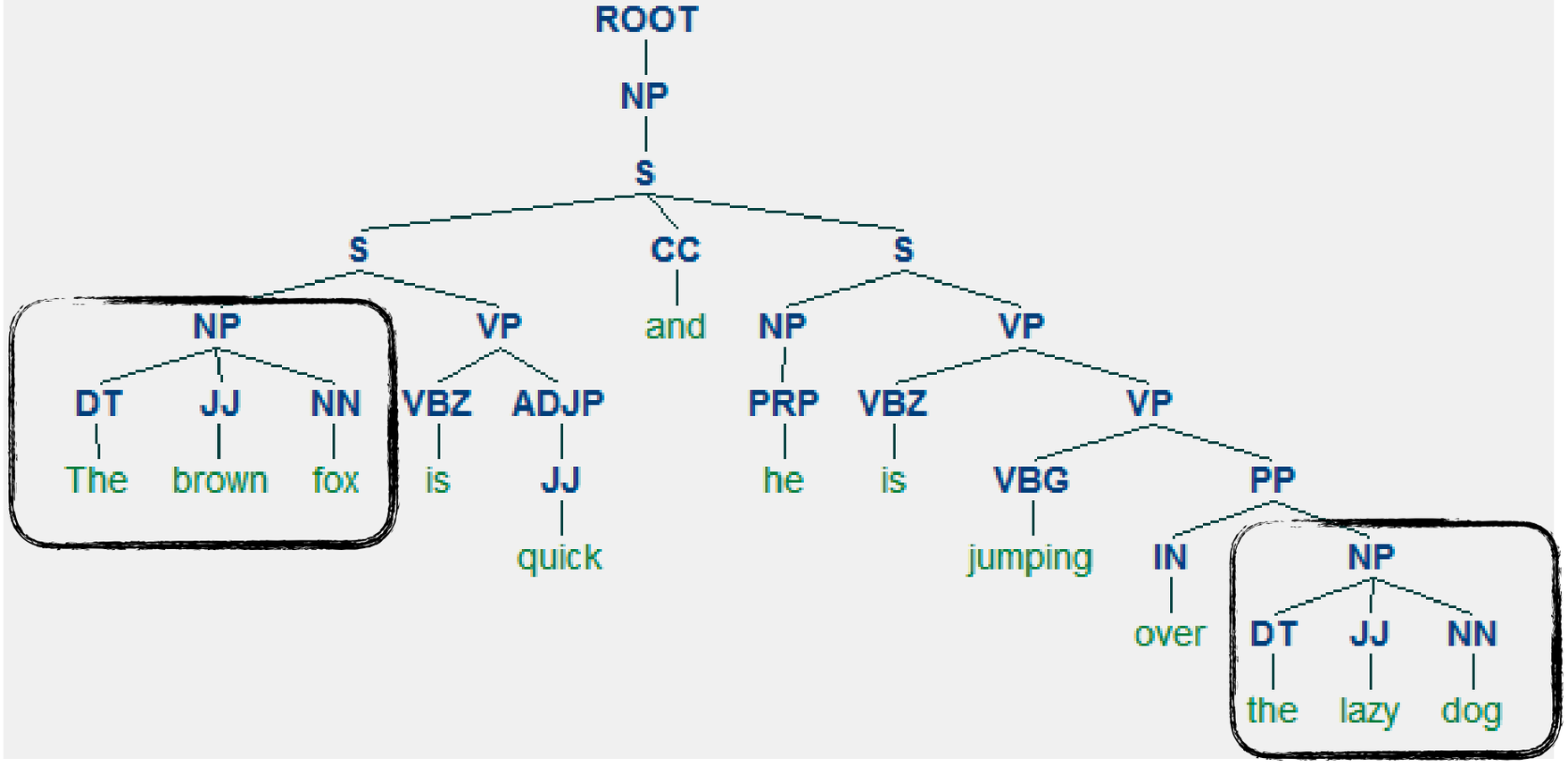
- Atomic units are words: **words** are assigned a category (part-of-speech). E.g., He-PRP, eats-VPZ, with-IN, meat-NN
- Words combine into phrases: with phrasal categories. E.g., PP -> IN NP (with meat)
- Phrases combine into bigger phrases (recursively) with phrasal categories. E.g., NP -> NP PP (spaghetti with meat)



Source: Stanford 224n

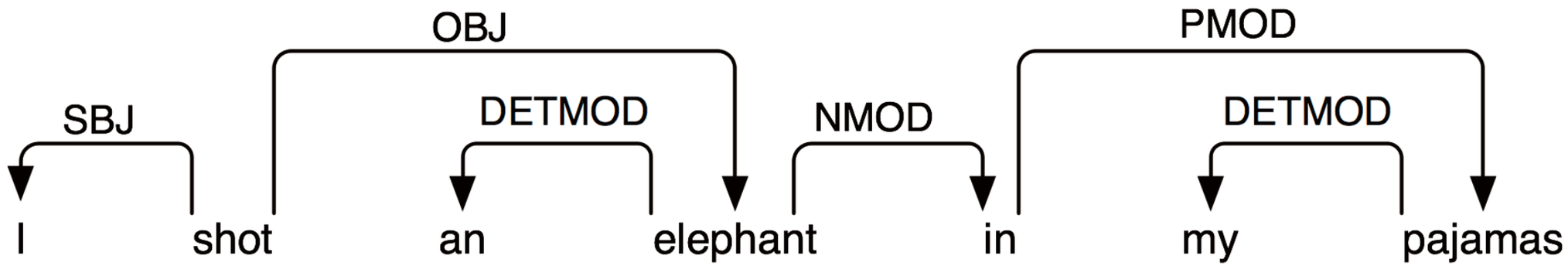
Constituency Structure

- Phrase structure organizes words into nested constituents
  - Can represent the process (grammar) with CFG rules
  - We can describe a language (e.g., English) using the CFG rules
  - Dominant structure in linguistics



Dependency Structure

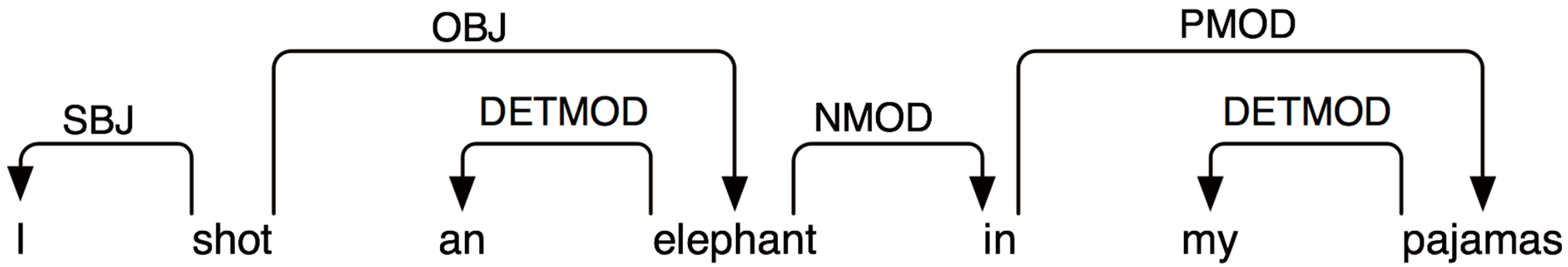
- Dependency structure shows which words depend on (modify or are arguments of) which other words
- Dependency is a binary asymmetric relation that holds between a **head** and its **dependents**.



Directed Acyclic Graph (Tree)

Source: NLTK

Dependency Structure



- Arrows point from heads to their dependents.
- Labels indicate the grammatical function of the dependent as subject, object or modifier
- Dependency grammar is used to directly express grammatical functions as a type of dependency.

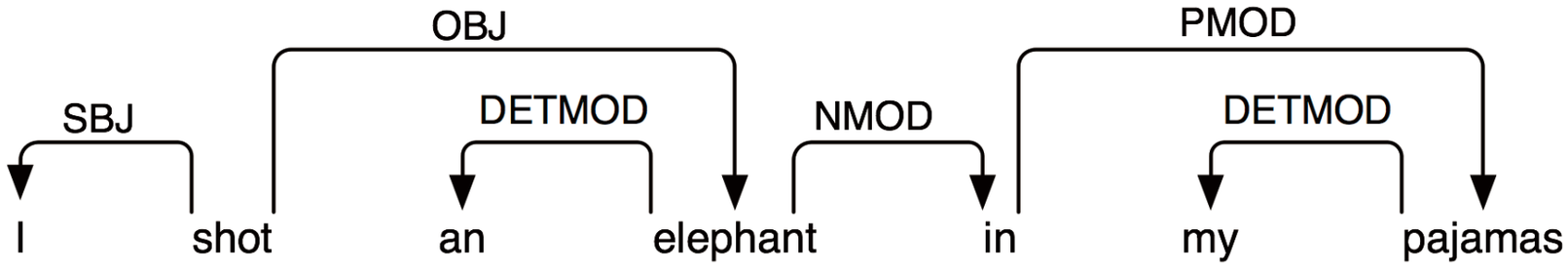
```
... 'shot' -> 'I' | 'elephant' | 'in'
... 'elephant' -> 'an' | 'in'
... 'in' -> 'pajamas'
... 'pajamas' -> 'my
```

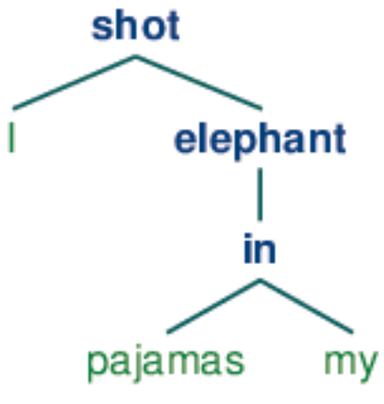
Source: NLTK

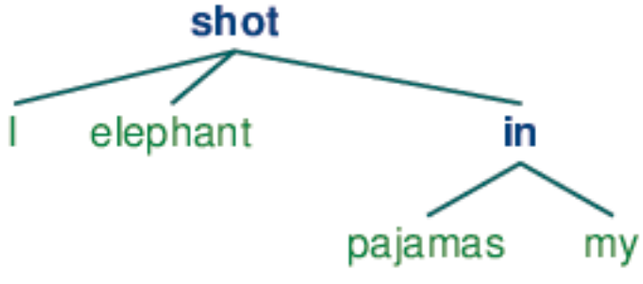
Dependency Structure

- A dependency graph is **projective** if, when the words are written in linear order, the edges can be drawn above the words without crossing.
- This is equivalent to saying that a word and all its descendents (dependents and dependents of its dependents, etc.) form a contiguous sequence of words within the sentence.

- We can parse most English sentences using a projective parser



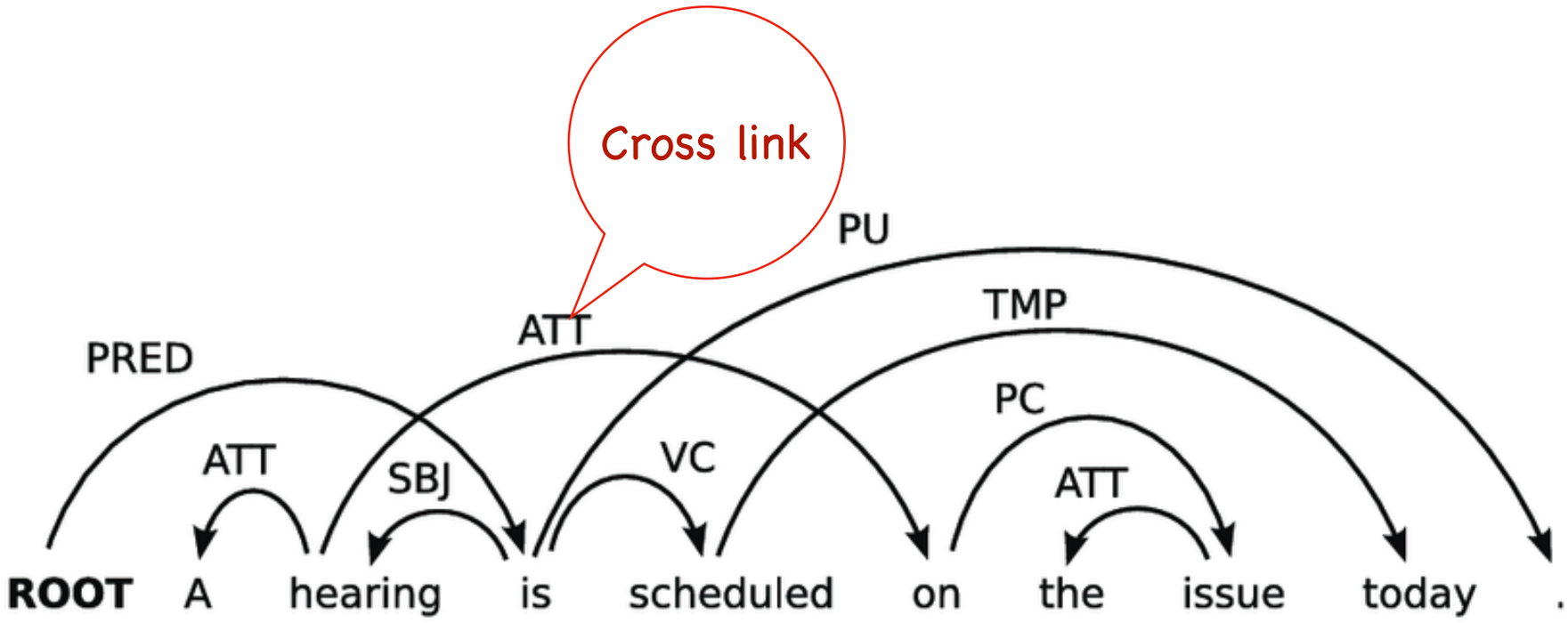




Source: NLTK

Dependency Structure

- Non-projective dependency structure



Source: Nivre (2010)

Why Syntactic Structures?

- To be able to interpret language correctly.
- Humans communicate complex ideas by composing words together into bigger units to convey complex meanings.
- We interpret the meaning of larger text units - entities, descriptive terms, facts, arguments, stories - by **semantic composition** of smaller elements
- We need to know what is connected to what to be able to understand bigger things from knowing about smaller parts

A **person on a snowboard** jumps into the air

Source: Stanford 224n

Why Syntactic Structures?

- PP attachment ambiguity

San Jose cops kill man with knife

Text Paper Close Translate Listen

San Jose cops kill man with knife

Source: Stanford 224n

Why Syntactic Structures?

- PP attachment ambiguity

Source: Stanford 224n

Why Syntactic Structures?

- PP attachment ambiguity

  - 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].

- # of possible structures: 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

Source: Stanford 224n

Why Syntactic Structures?

- **Coordination scope ambiguity**

Small rats and mice can squeeze into holes or cracks in the wall.

Shuttle veteran and longtime NASA executive Fred Gregory appointed to board

Why Syntactic Structures?

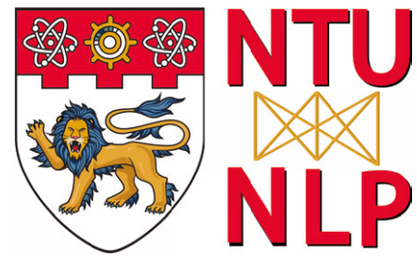
- **Adjectival modifier ambiguity**

Students get their first lab experience

Why Syntactic Structures?

- Verb Phrase attachment

Are Languages Hierarchical?



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]

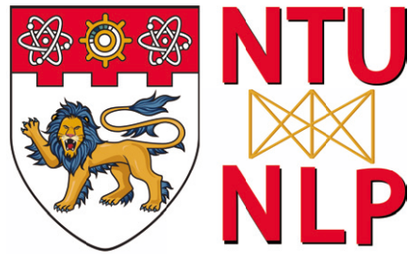
+ See all authors and affiliations

Science 22 Nov 2002:
Vol. 298, Issue 5598, pp. 1569-1579
DOI: 10.1126/science.298.5598.1569

Abstract

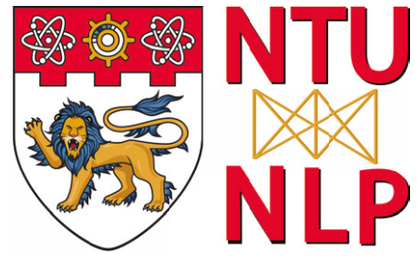
We argue that an understanding of the faculty of language requires substantial interdisciplinary cooperation. We suggest how current developments in linguistics can be profitably wedded to work in evolutionary biology, anthropology, psychology, and neuroscience. We submit that a distinction should be 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).

Are Languages Hierarchical?



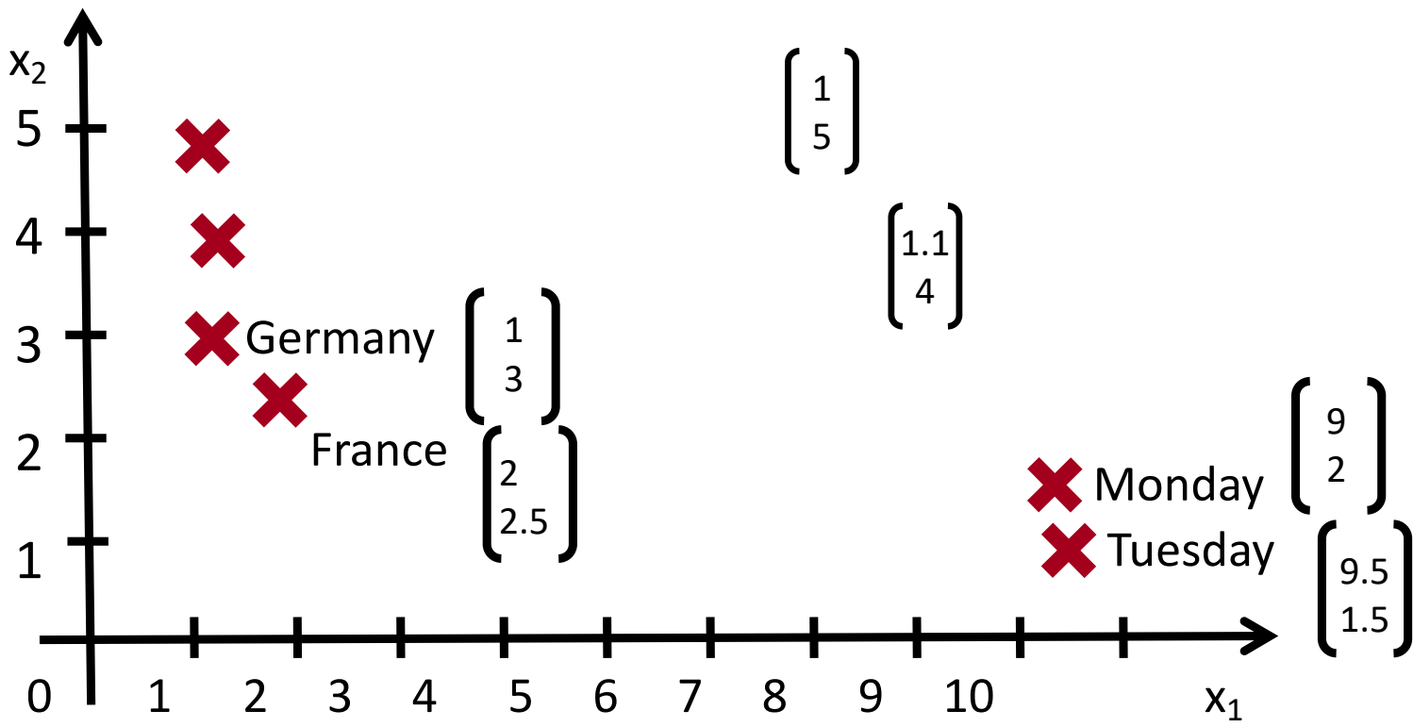
- Cognitively somewhat debatable (need to head to infinity)
- But: recursion is natural for describing language
  - [The person standing next to [the man from [the company that purchased [the firm that you used to work at]]]]
  - NP containing a NP containing a NP
- It's a very powerful prior for language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

Today's Plan



- Syntactic structures
  - Constituency & Dependency structures
- Need for syntactic structures
- Are languages hierarchical?
- Parsing with Recursive Neural Networks
- Backpropagation through tree
- Semantic composition with Tree RNNs
- Recent Parsers

Building on Word Vectors



- How can we represent the meaning of longer phrases
  - map them into the same vector space!

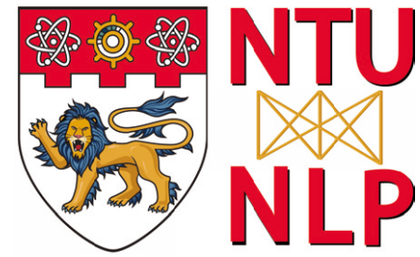
My birth place

the place where I was born

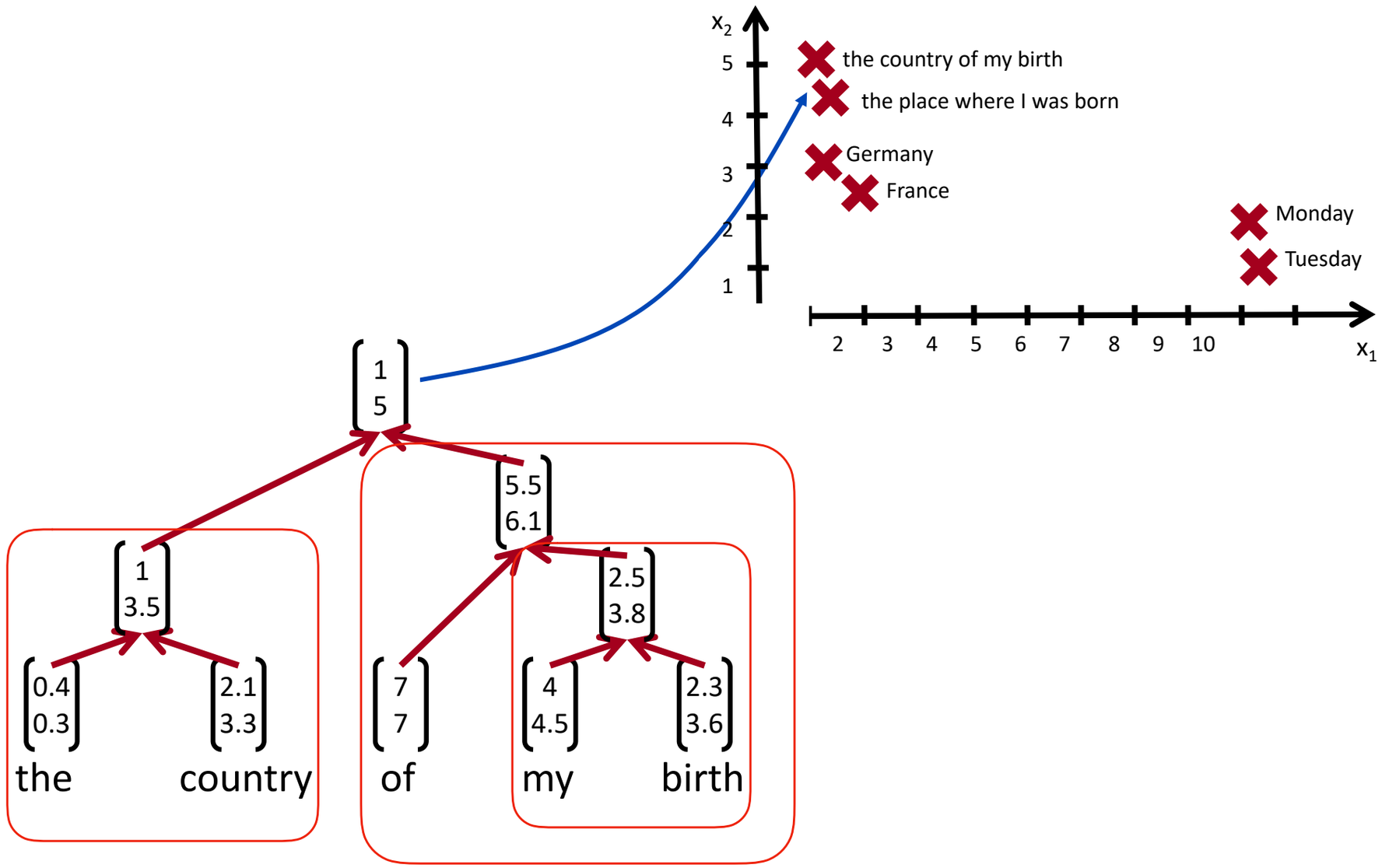
The place of my birth

Source: Stanford 224n

How Should We Map Phrases



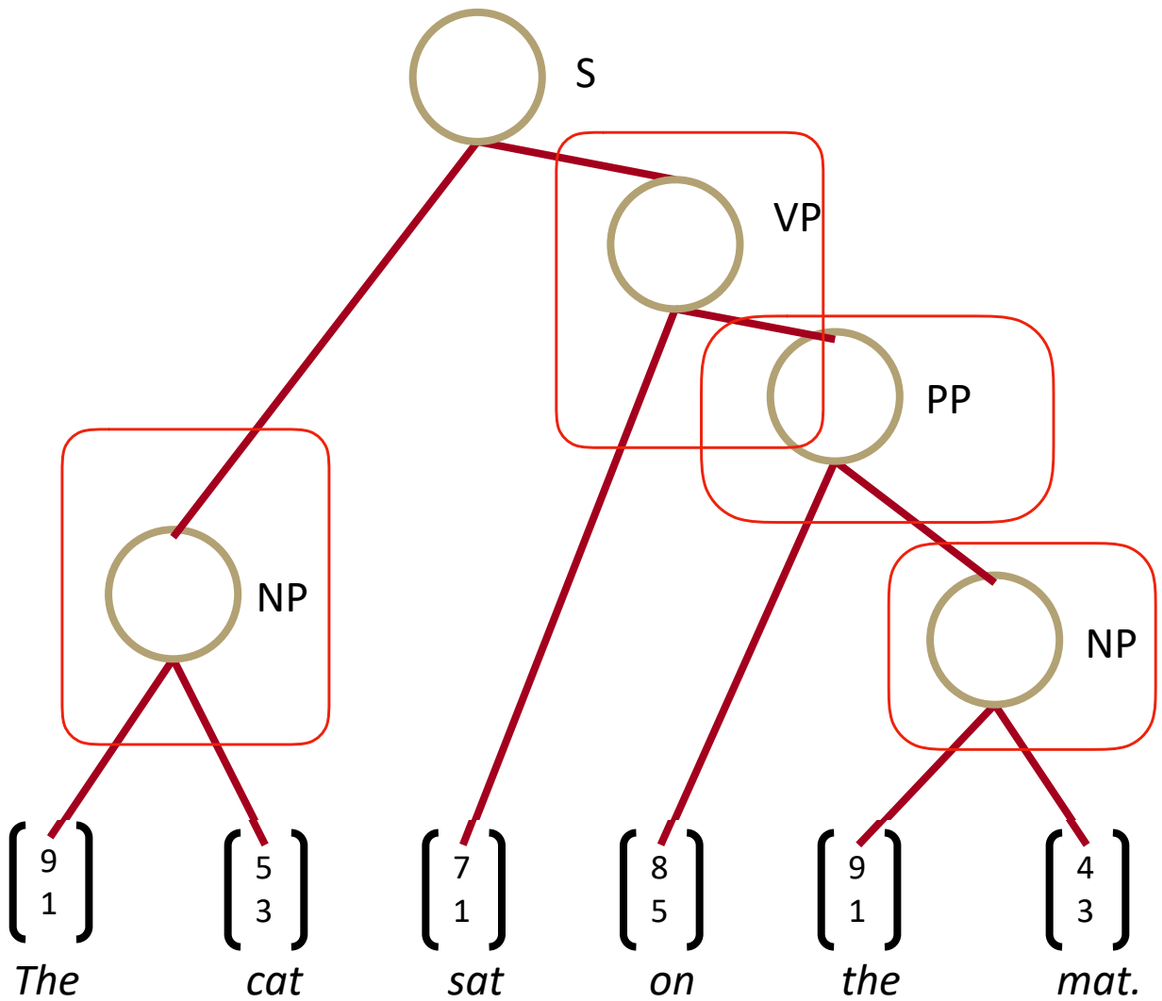
- Map them into the same vector space!



The meaning (vector) of a sentence is determined by
(1) the meanings of its words and (2) the rules that combine them.

Source: Stanford 224n

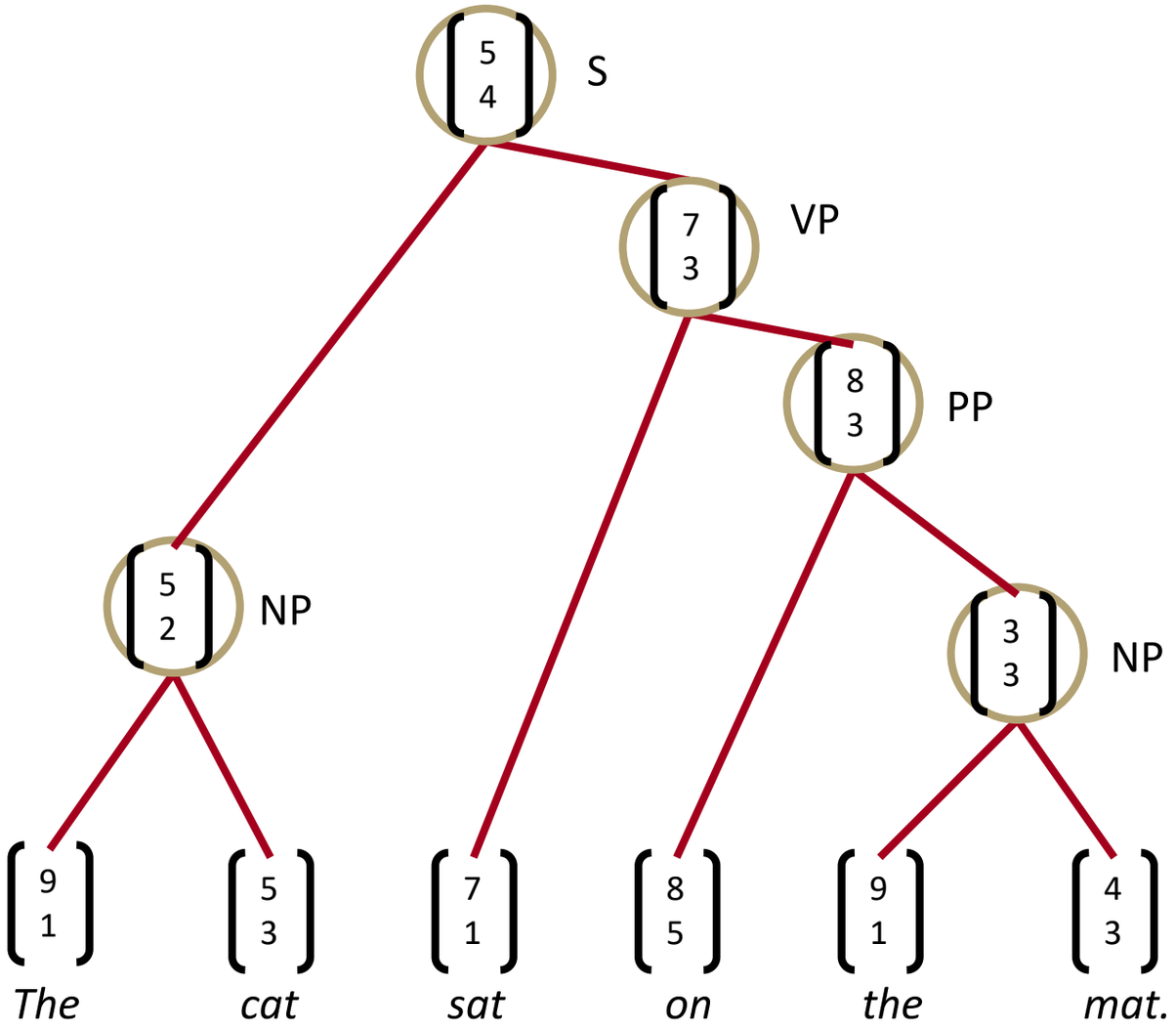
Constituency Parsing



Jointly learn **parse trees** and compositional vector representations

Source: Stanford 224n

Constituency Parsing



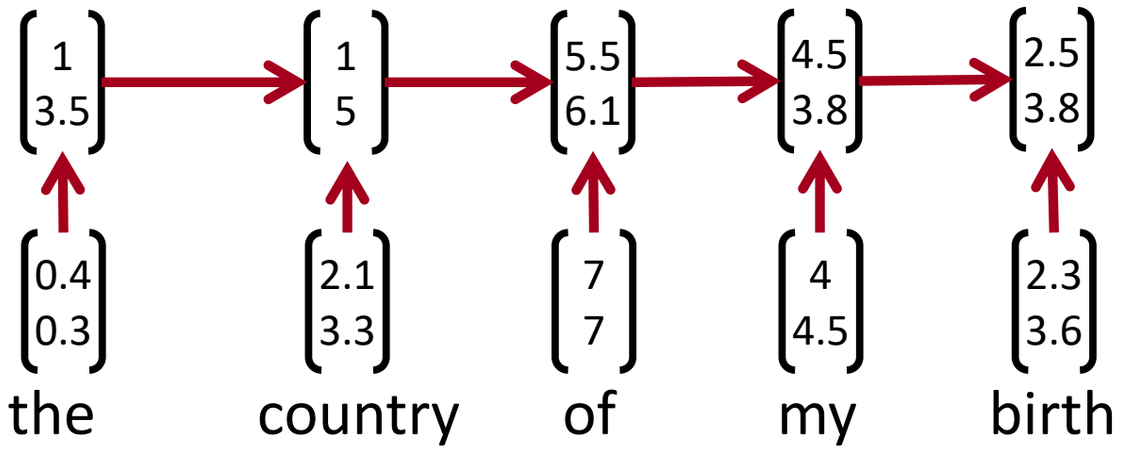
Jointly learn parse trees and **compositional vector representations**

Source: Stanford 224n

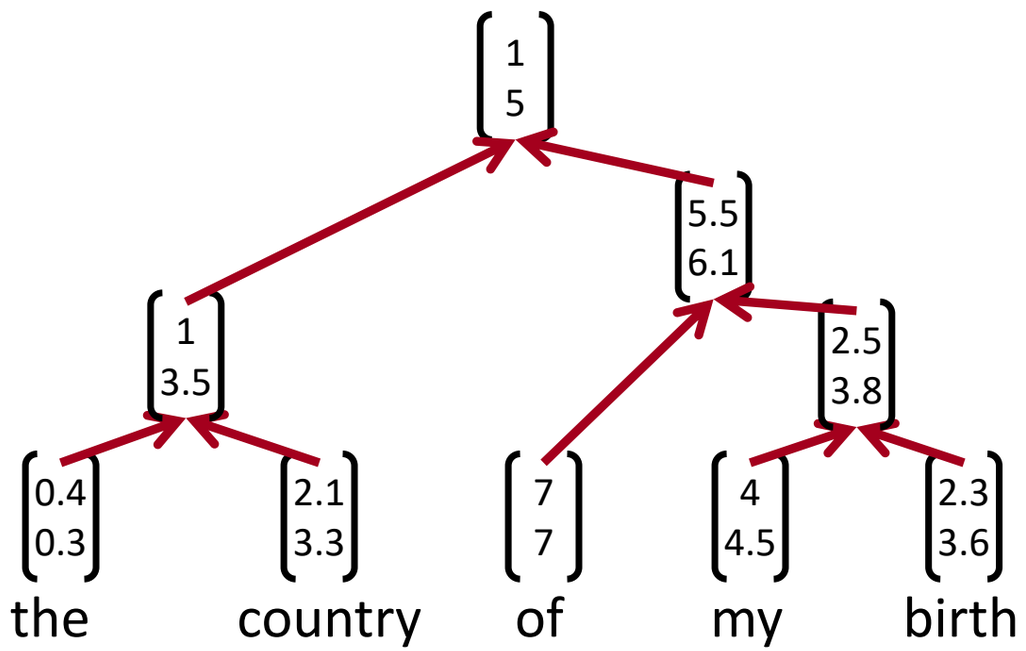
Recursive vs. Recurrent

NTU NLP

Recurrent



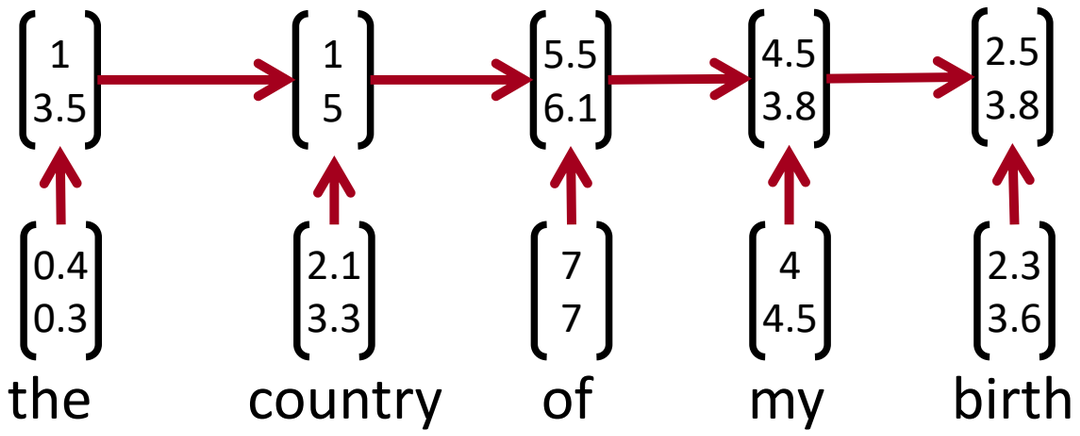
Recursive



Source: Stanford 224n

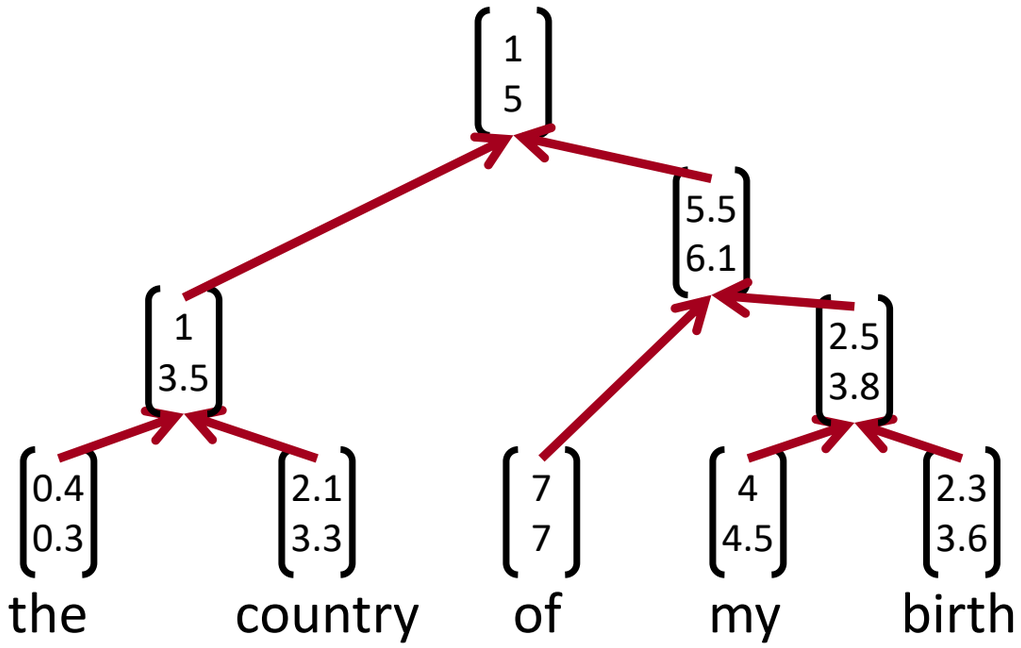
Recursive vs. Recurrent

Recurrent



Cannot capture phrases without prefix context and often capture too much of last words in final vector

Recursive



Require a tree structure to get phrase representations.

Source: Stanford 224n

Recursive Nets for Structure Prediction

Inputs: two candidate children's representations

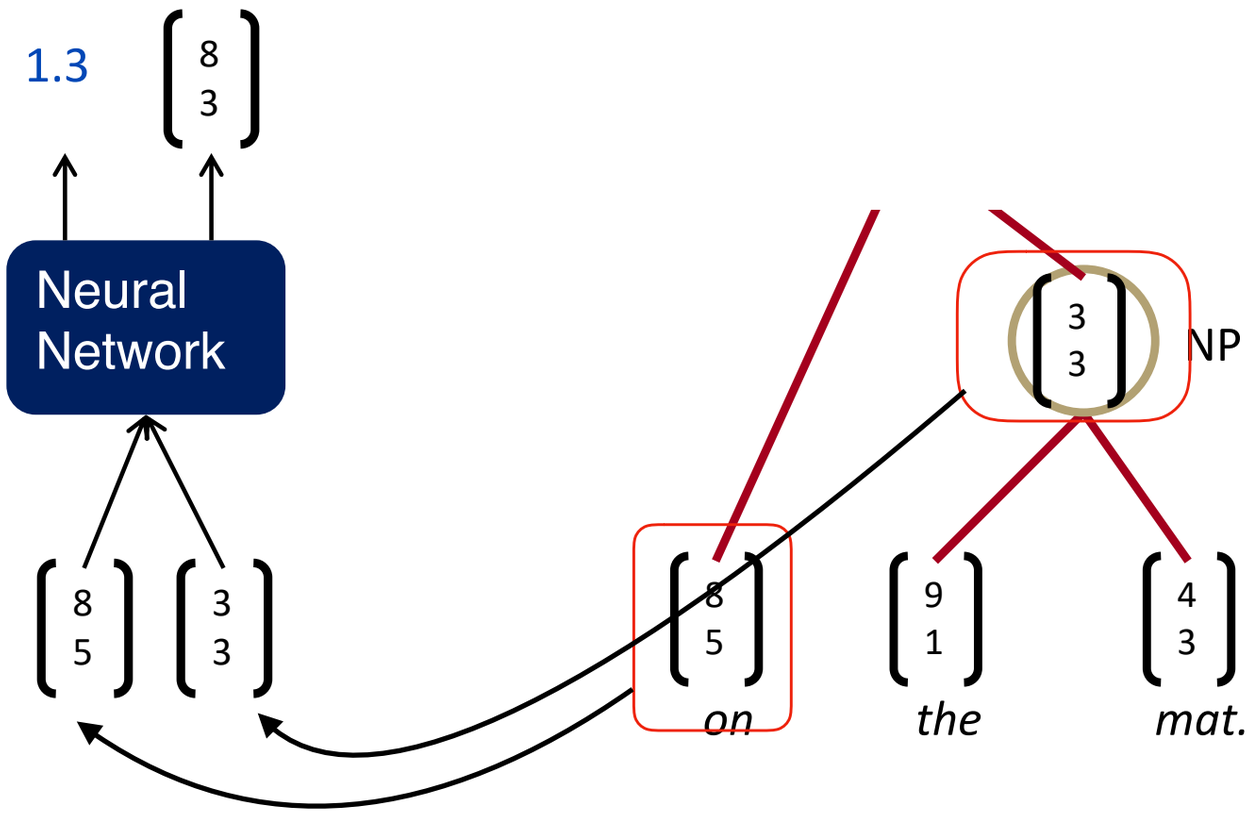
Outputs:

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

score = $U^T p$

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

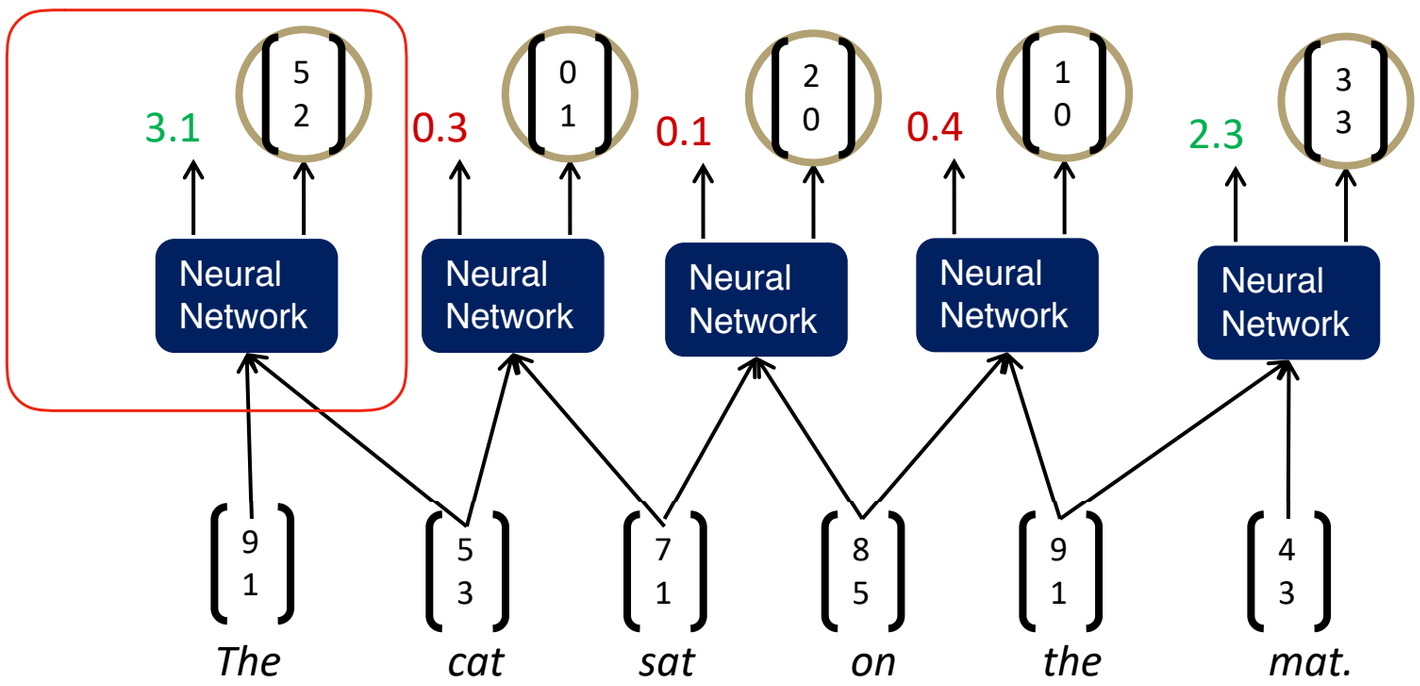
Same W parameters at all nodes of the tree



Source: Stanford 224n

Parsing a Sentence

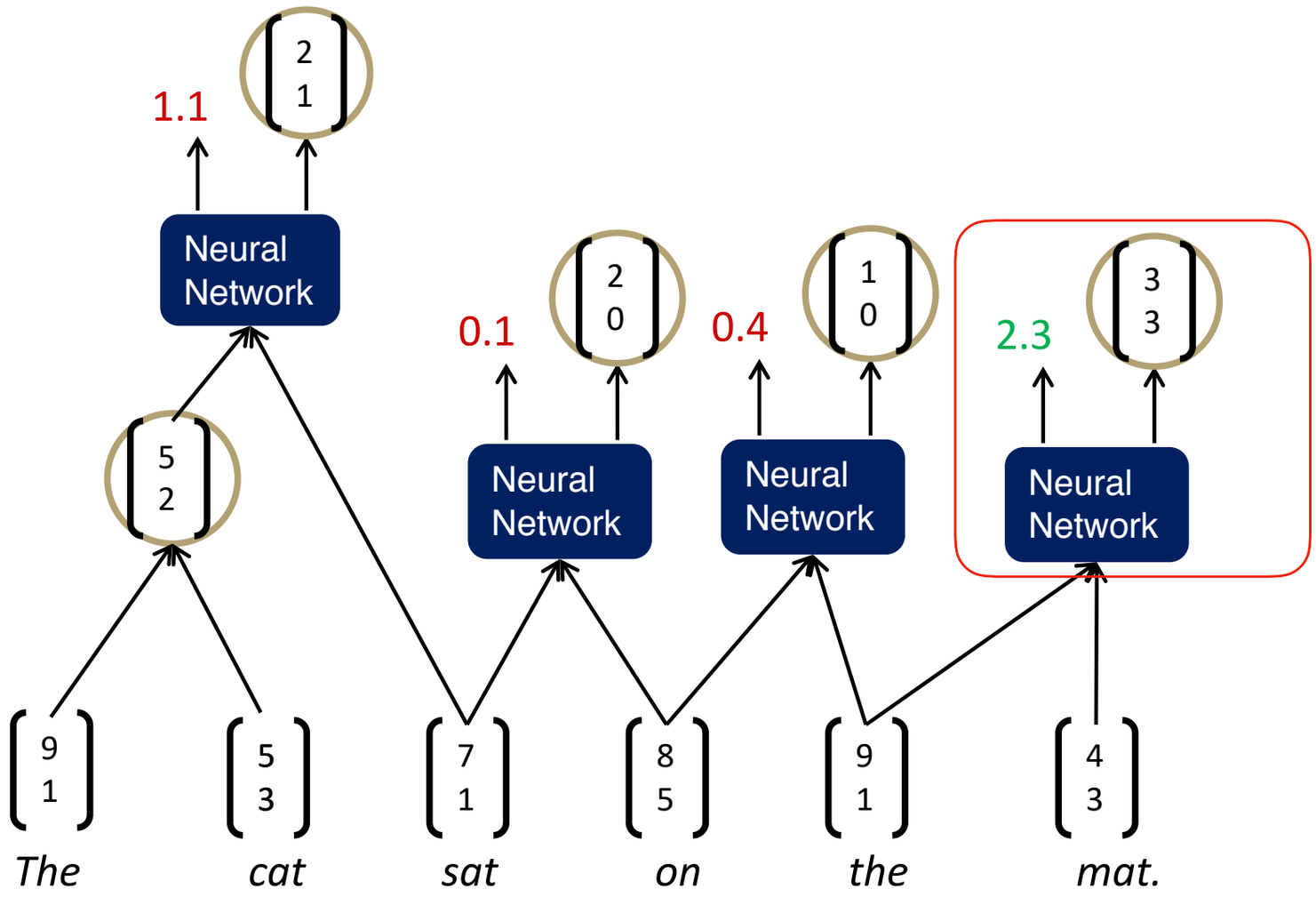
Greedy Approach



Source: Stanford 224n

Parsing a Sentence

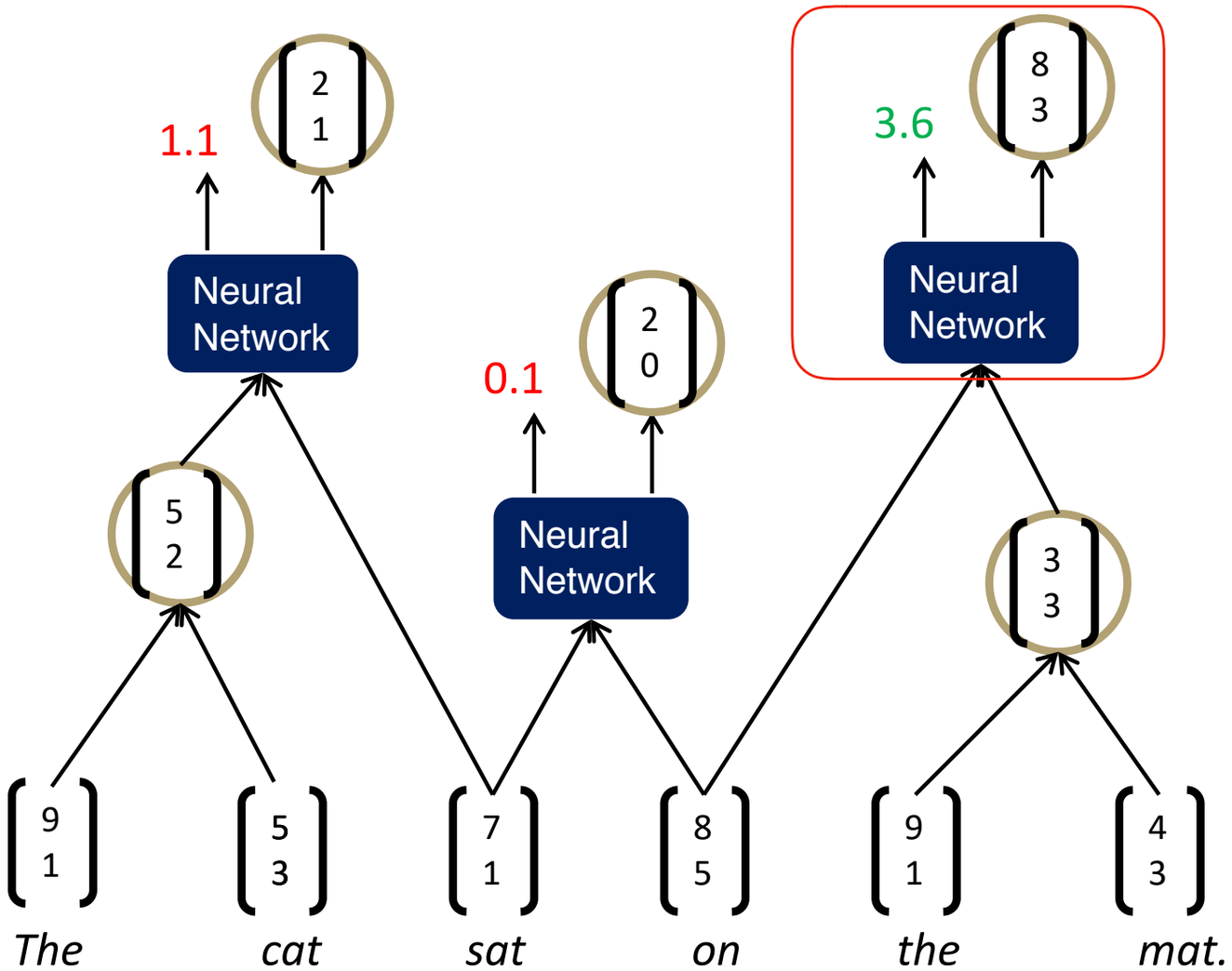
Greedy Approach



Source: Stanford 224n

Parsing a Sentence

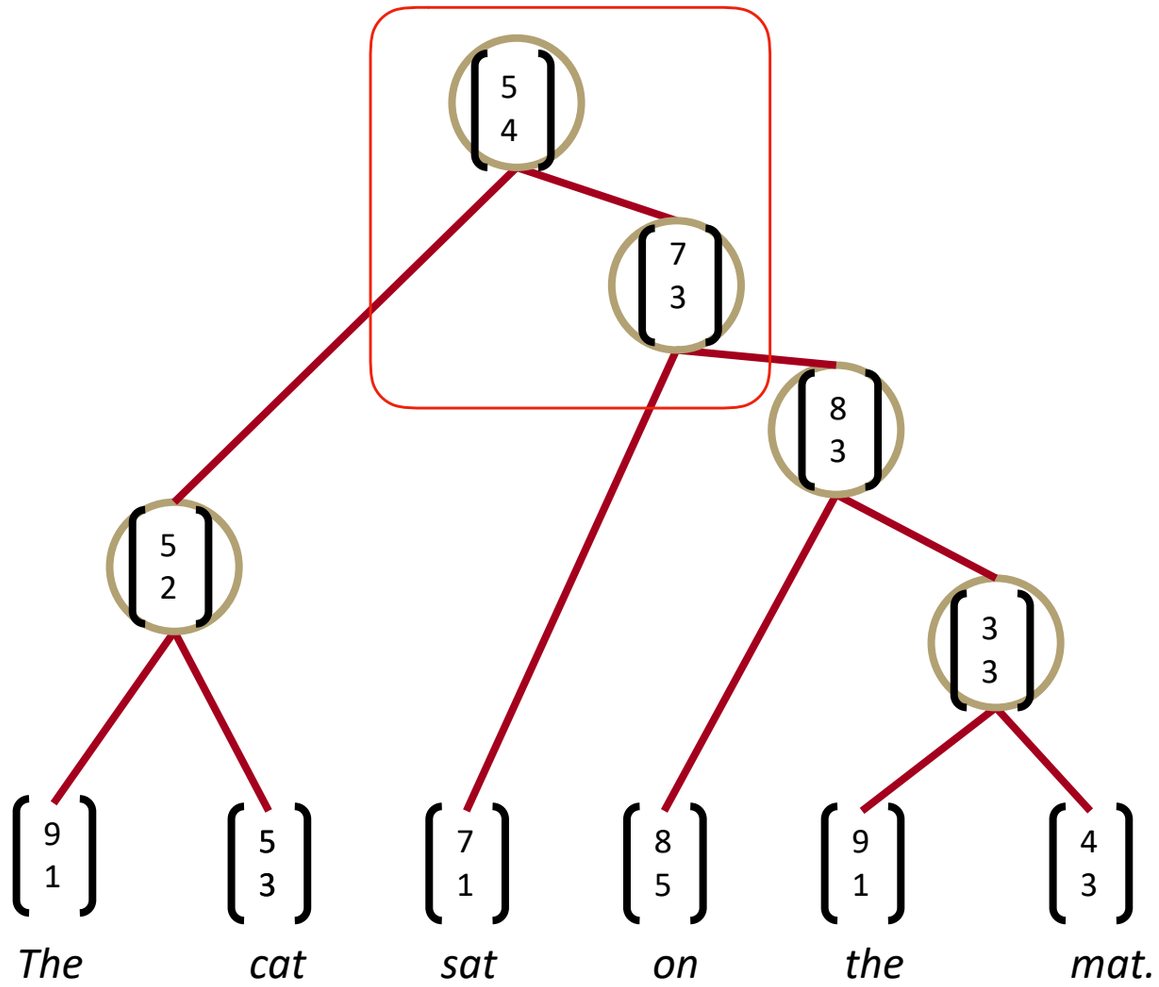
Greedy Approach



Source: Stanford 224n

Parsing a Sentence

Greedy Approach



Source: Stanford 224n

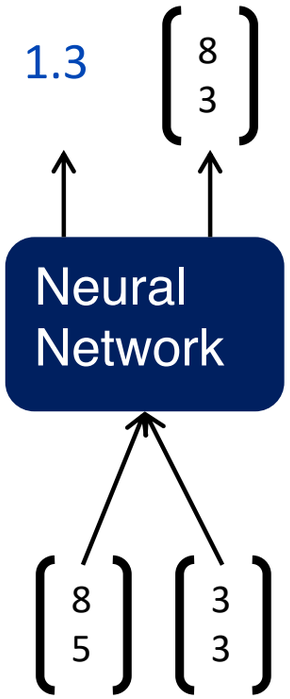
Parsing a Sentence

Max-margin Training

- 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



Parsing a Sentence

NTU NLP

Max-margin Training

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

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

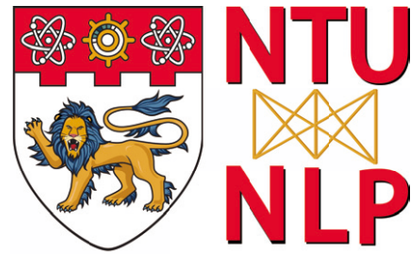
Beam search with chart (instead of Greedy in previous slides)

penalizes all incorrect decisions

- Cannot do CKY (Dynamic Programming)

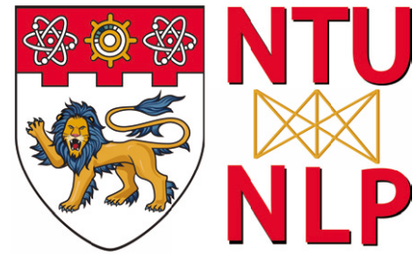
Source: Stanford 224n

Today's Plan



- Syntactic structures
  - Constituency & Dependency structures
- Need for syntactic structures
- Are languages hierarchical?
- Parsing with Recursive Neural Networks
- Backpropagation through tree
- Semantic composition with Tree RNNs
- Recent Parsers

Backpropagation through tree



Same backpropagation as before

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

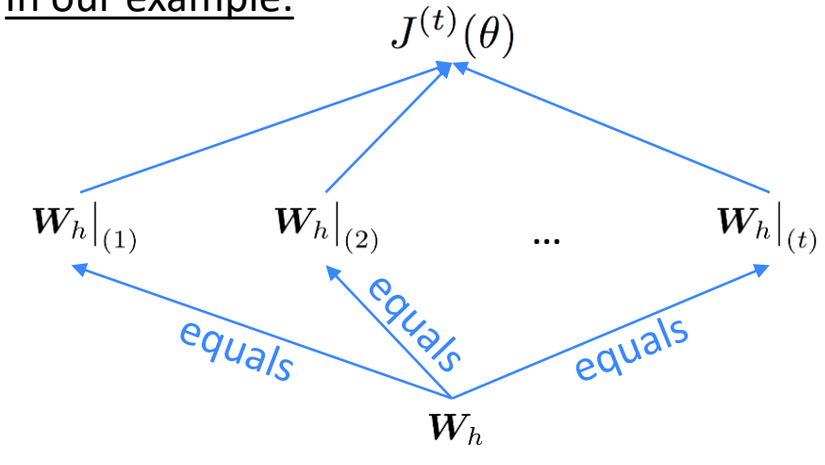
Source: Stanford 224n

Backpropagation through tree

Same backpropagation as before

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

In our example:



Apply the multivariable chain rule:

$$\frac{\partial J^{(t)}}{\partial \boldsymbol{W}_h} = \sum_{i=1}^{t} \frac{\partial J^{(t)}}{\partial \boldsymbol{W}_h}\bigg|_{(i)} \boxed{\frac{\partial \boldsymbol{W}_h|_{(i)}}{\partial \boldsymbol{W}_h}} \quad (= 1)$$

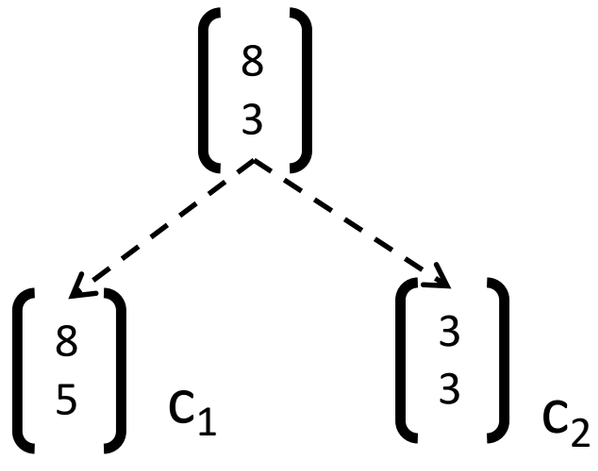
$$= \sum_{i=1}^{t} \frac{\partial J^{(t)}}{\partial \boldsymbol{W}_h}\bigg|_{(i)}$$

Source: Stanford 224n

Backpropagation through tree

Same backpropagation as before

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

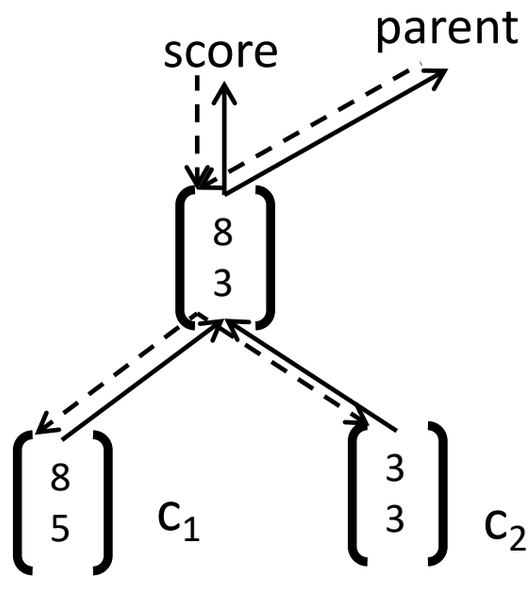


Source: Stanford 224n

Backpropagation through tree

Same backpropagation as before

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



Source: Stanford 224n

Python Code

```
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)
```

Source: Stanford 224n

Python Code

```
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
```

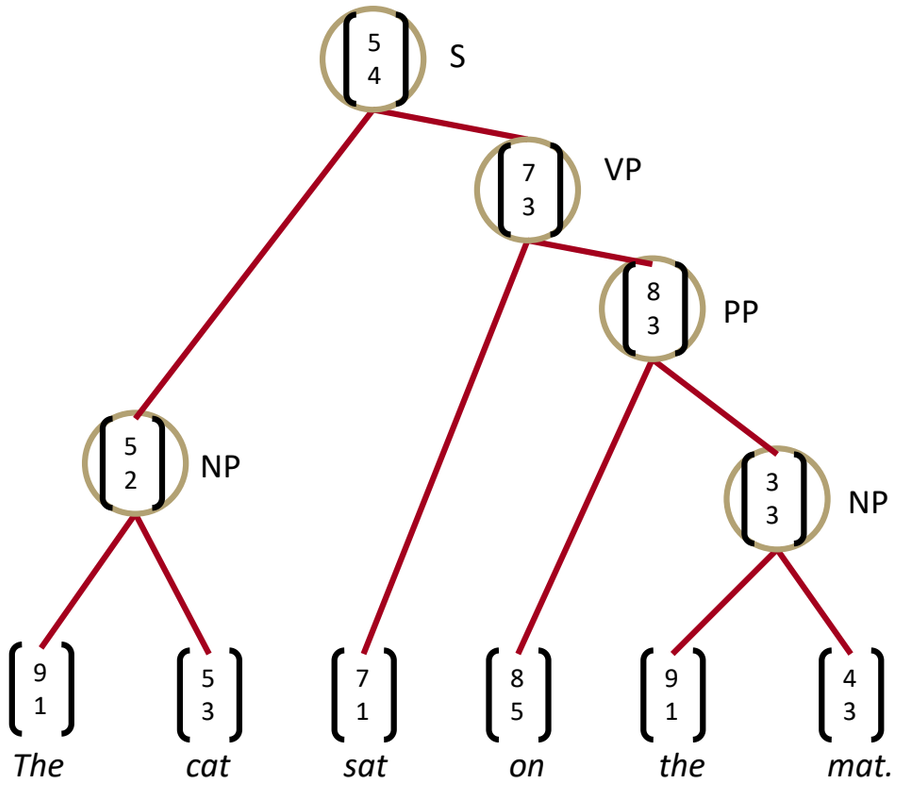
```
# f'(z) now:
deltas *= (node.h != 0)
```

```
# 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:])
```

Source: Stanford 224n

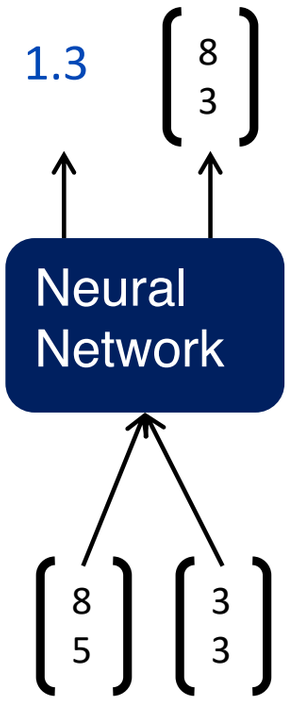
Limitations of Basic Tree-RNNs



score = $U^T p$

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

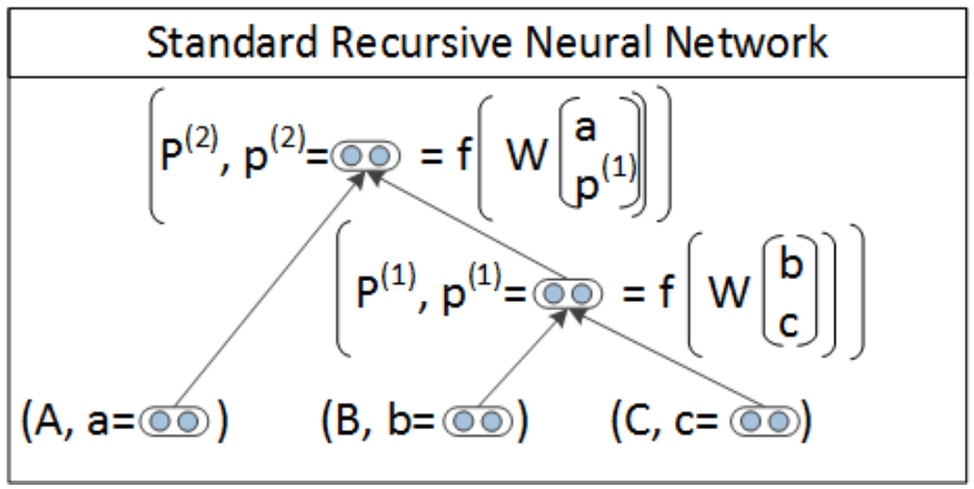
Same W parameters at all nodes of the tree

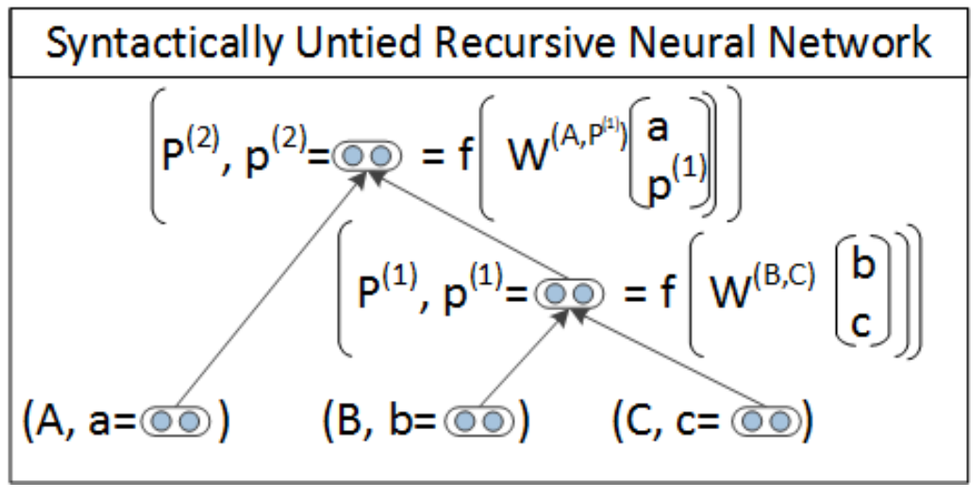


- Single weight matrix TreeRNN could capture some phenomena but not adequate for more complex, higher order composition and parsing long sentences
- There is no real interaction between the input words
- The composition function is the same for all syntactic categories, punctuation, etc

Source: Stanford 224n

Version 2: Syntactically-Untied RNN

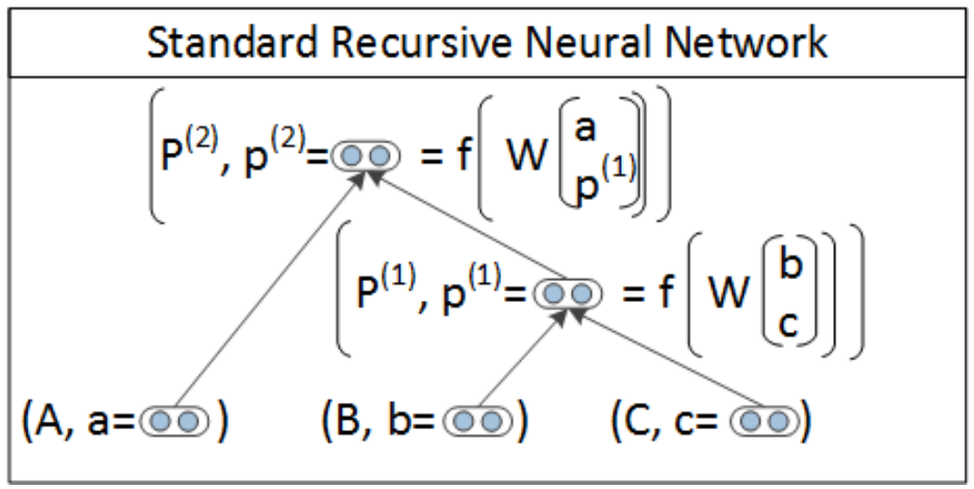


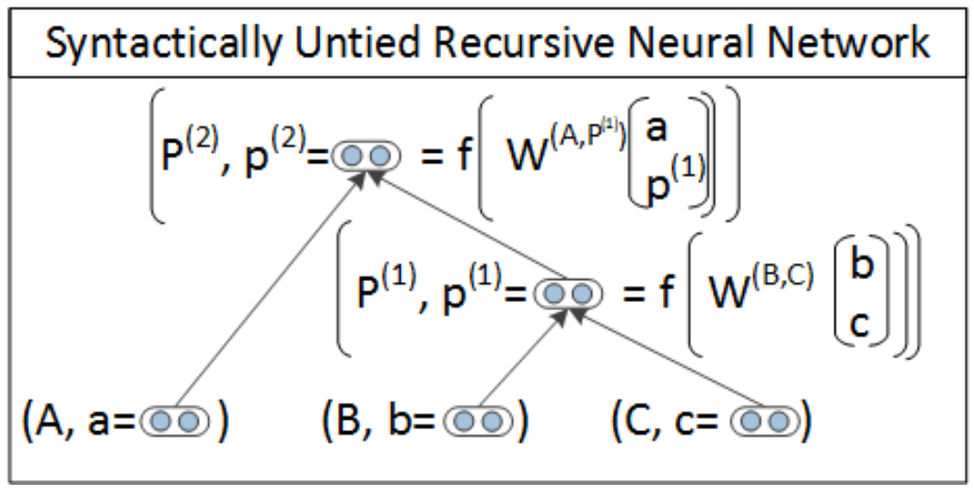


- A symbolic Context-Free Grammar (CFG) backbone is adequate for basic syntactic structure
- We use the discrete syntactic categories of the children to choose the composition matrix
- A TreeRNN can do better with different composition matrix for different syntactic environments
- The result gives us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

Source: Stanford 224n

Version 2: Syntactically-Untied RNN

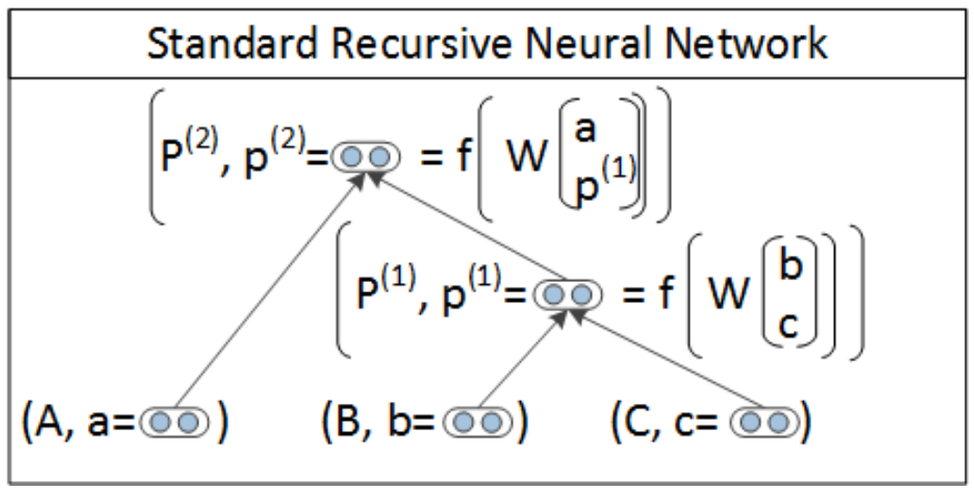


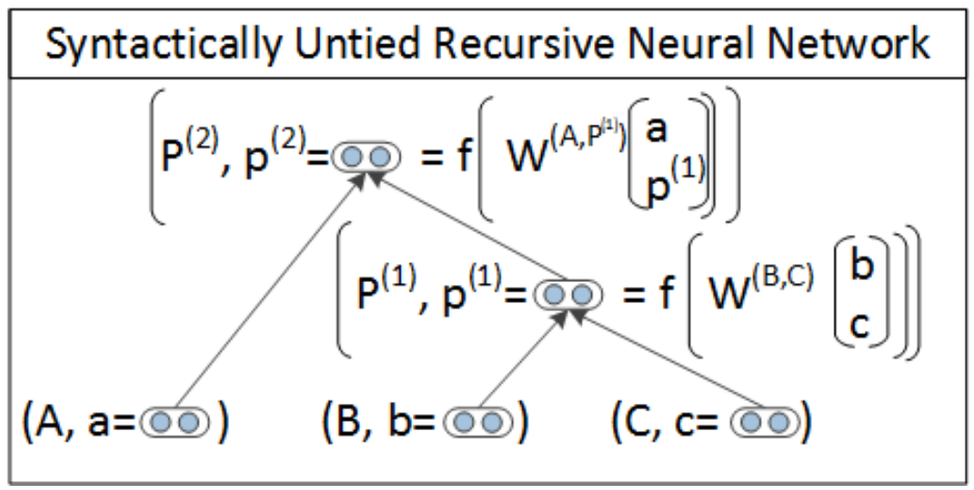


- Problem: Speed. Every candidate score in beam search needs a matrix-vector product.
- Solution: Compute score only for a subset of trees coming from a simpler, faster model (PCFG)
  - Prunes very unlikely candidates for speed
  - Provides coarse syntactic categories of the children for each beam candidate
- Compositional Vector Grammar = PCFG + TreeRNN

Source: Stanford 224n

Version 2: Syntactically-Untied RNN





- Compositional Vector Grammar = PCFG + TreeRNN
- Scores at each node computed by combination of PCFG and SU-RNN:

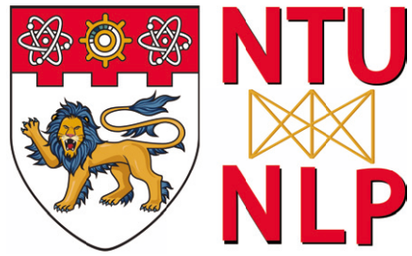
$$s\left(p^{(1)}\right) = \left(v^{(B,C)}\right)^T p^{(1)} + \log P(P_1 \rightarrow B \;\; C)$$

- Factoring discrete and continuous parsing in one model:

$$P((P_1, p_1) \rightarrow (B, b)(C, c))$$
$$= P(p_1 \rightarrow b \;\; c | P_1 \rightarrow B \;\; C) P(P_1 \rightarrow B \;\; C)$$

Source: Stanford 224n

Parsing Results



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

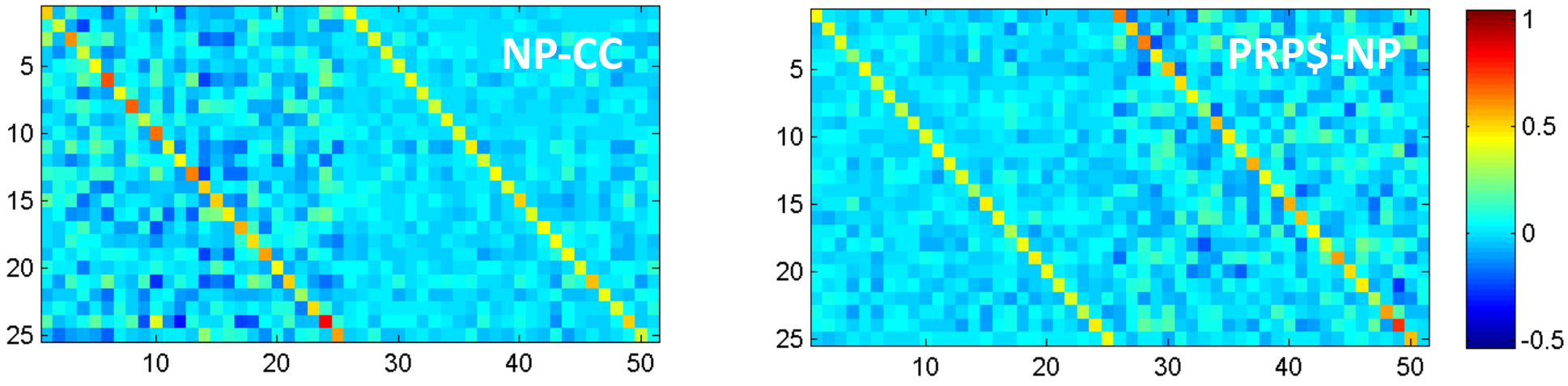
- Standard WSJ split, labeled F1
- Based on simple PCFG with fewer states
- Fast pruning of search space, few matrix-vector products
- 3.8% higher F1

Source: Stanford 224n

Analysis of Learned Representations

NTU NLP

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

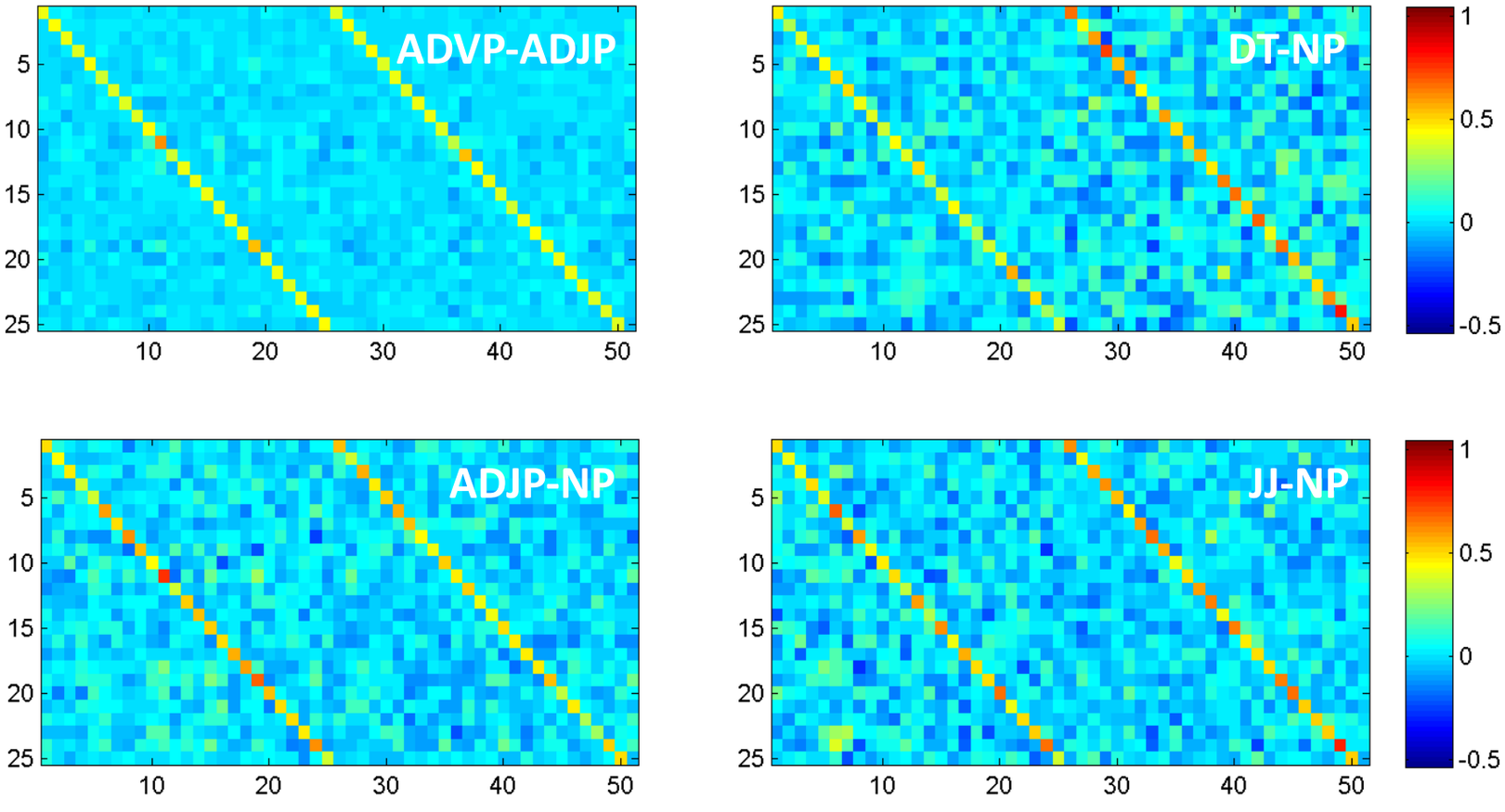


- Learns soft notion of head words

Source: Stanford 224n

Analysis of Learned Representations

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



- Learns soft notion of head words

Source: Stanford 224n

Analysis of Learned 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.

Source: Stanford 224n

Version 3: Compositionality Through Recursive Matrix-Vector Spaces

So far:

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

We untied the weights in Version 2 to make the composition function more powerful

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

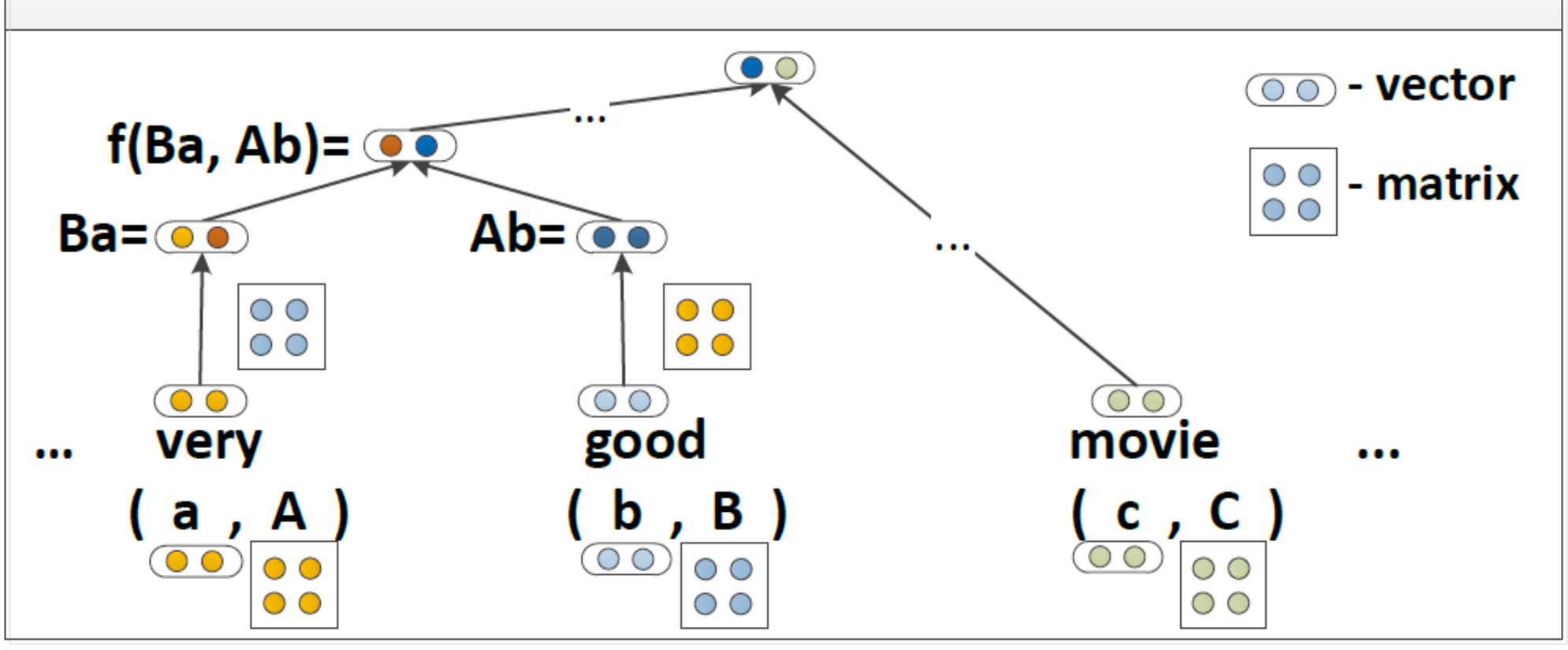
Source: Stanford 224n

Version 3: Compositionality Through Recursive Matrix-Vector Spaces

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

$$p = \tanh\left(W\begin{bmatrix}C_2c_1\\C_1c_2\end{bmatrix} + b\right)$$

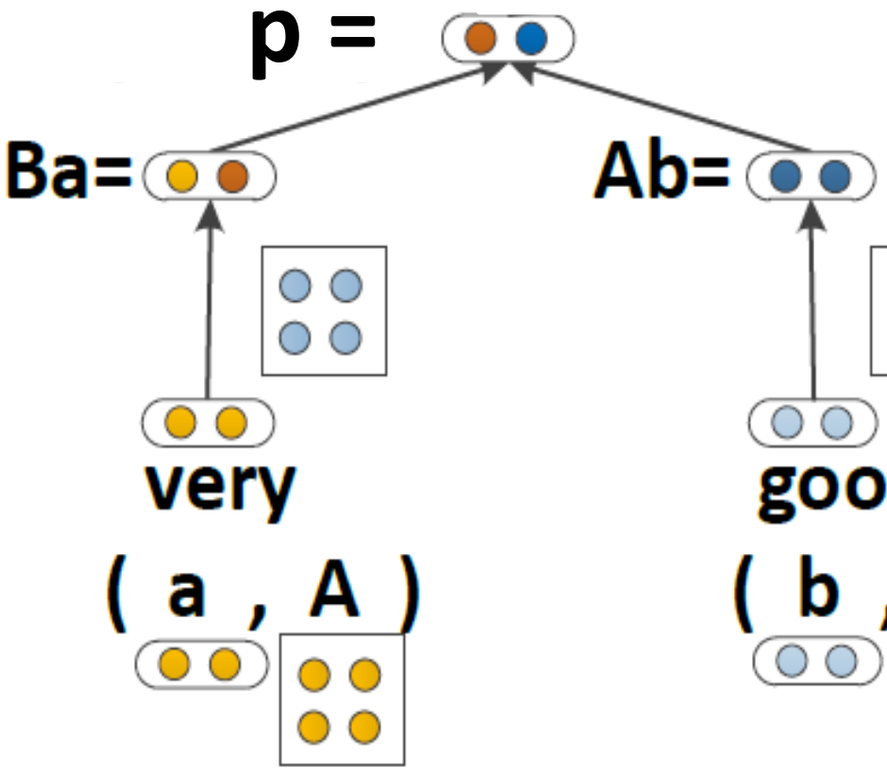
Recursive Matrix-Vector Model



Source: Stanford 224n

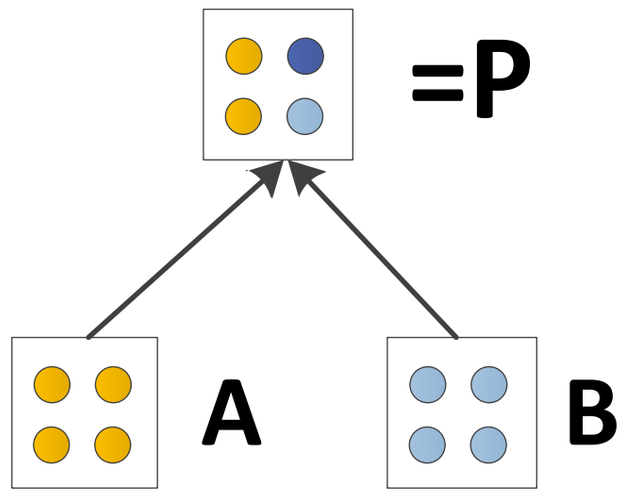
Version 3: Compositionality Through Recursive Matrix-Vector Spaces

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



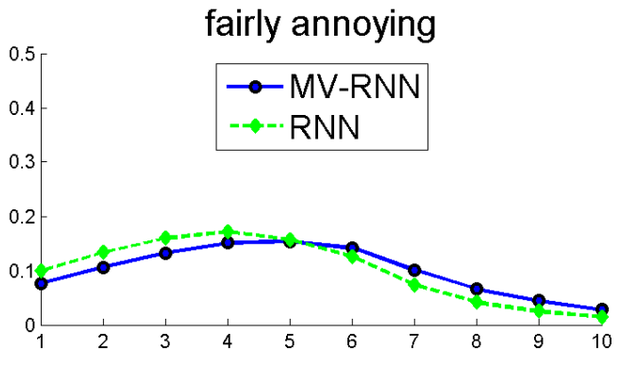
$$P = g(A, B) = W_M \begin{bmatrix} A \\ B \end{bmatrix}$$

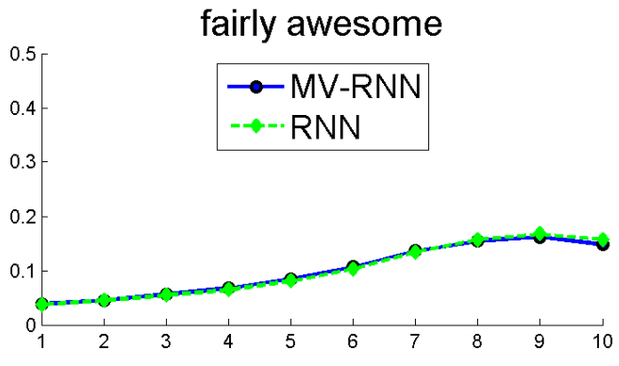
$$W_M \in \mathbb{R}^{n \times 2n}$$

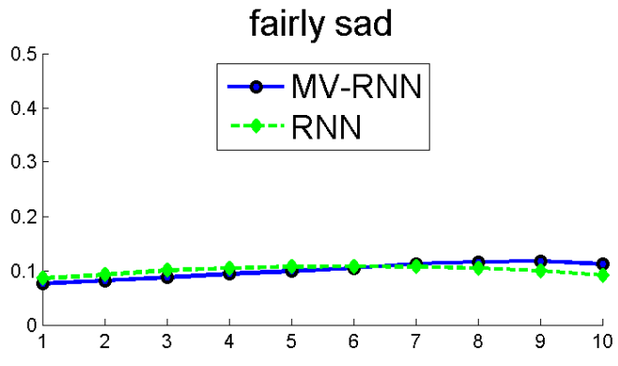


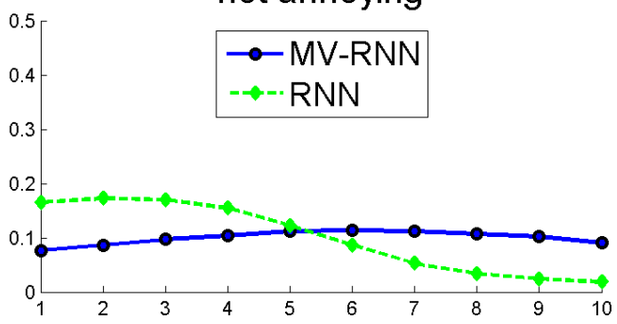
Source: Stanford 224n

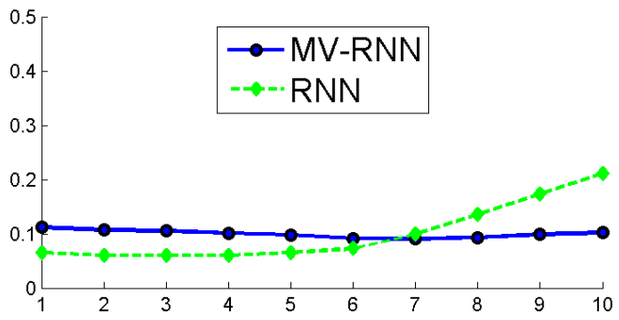
Predicting Sentiment Distributions

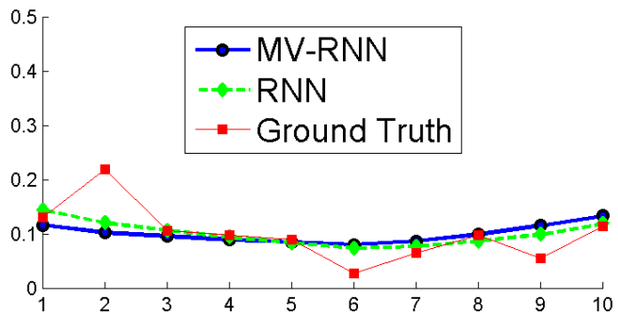


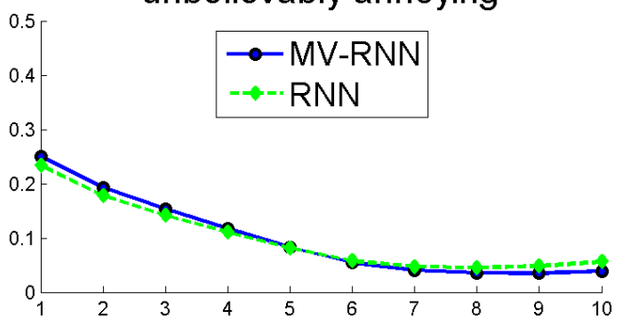


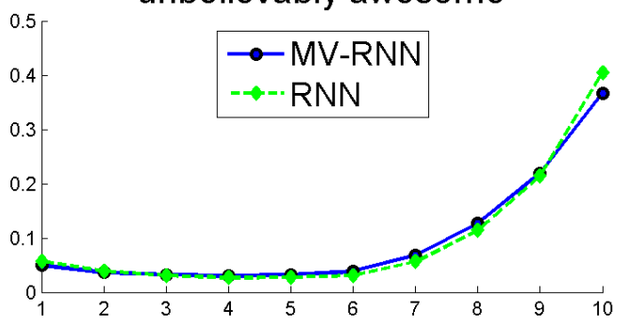


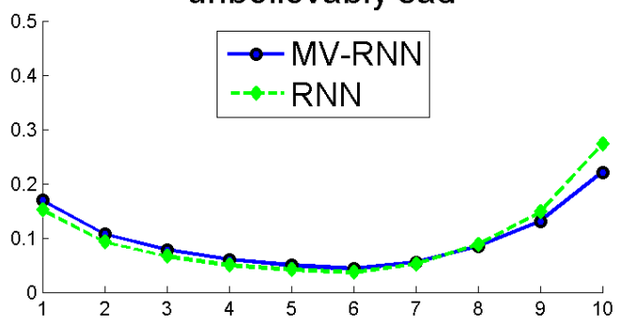






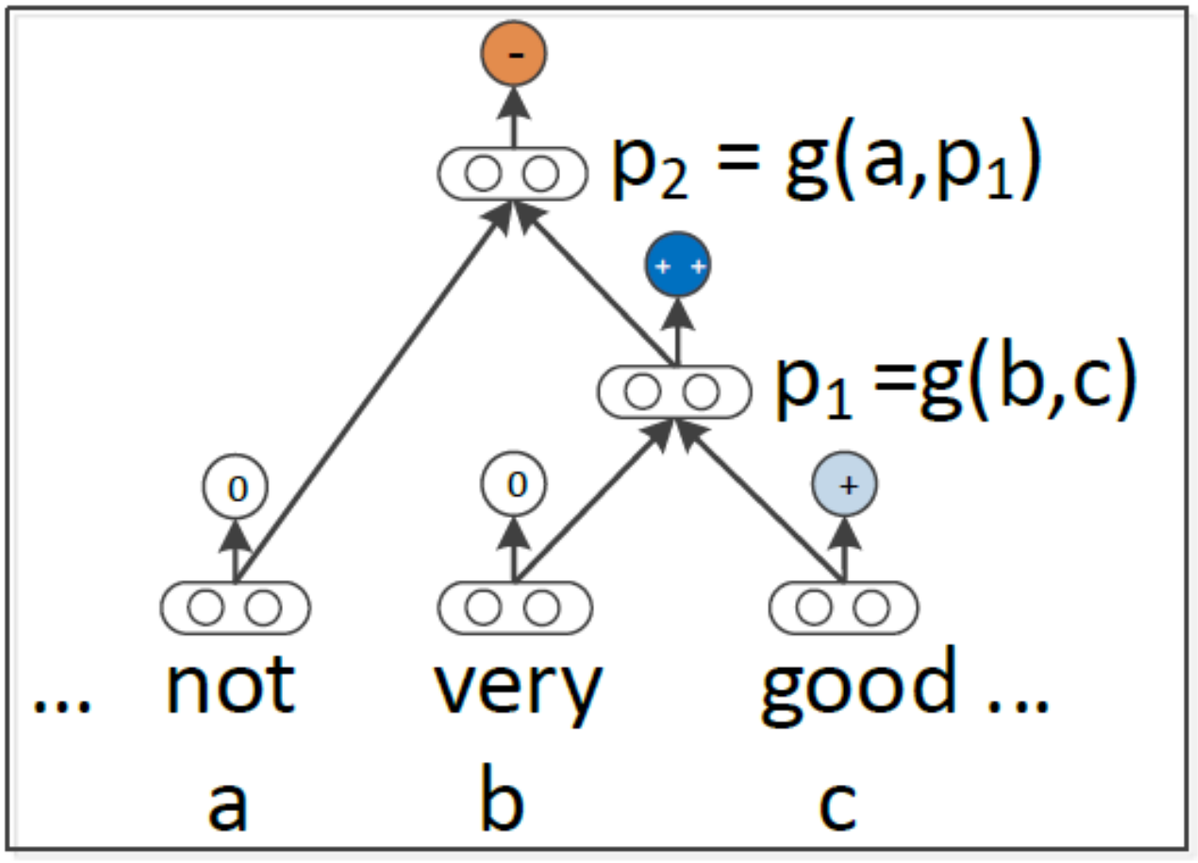


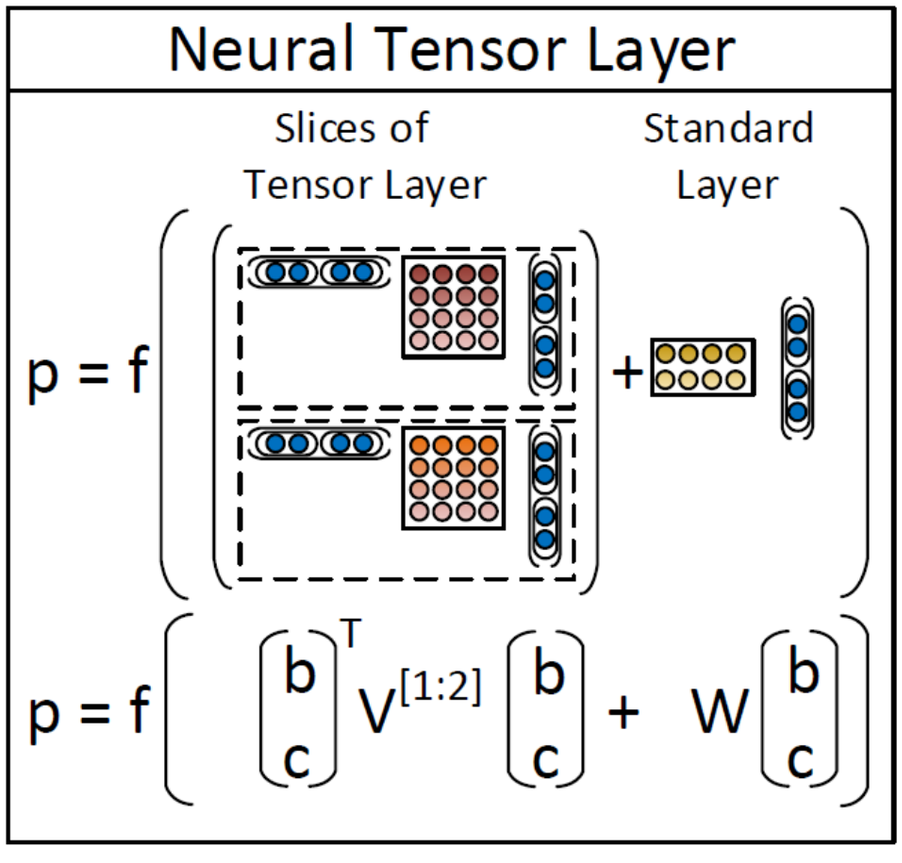




Source: Stanford 224n

Version 4: Recursive Neural Tensor Network





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

Source: Stanford 224n

Sentiment Analysis

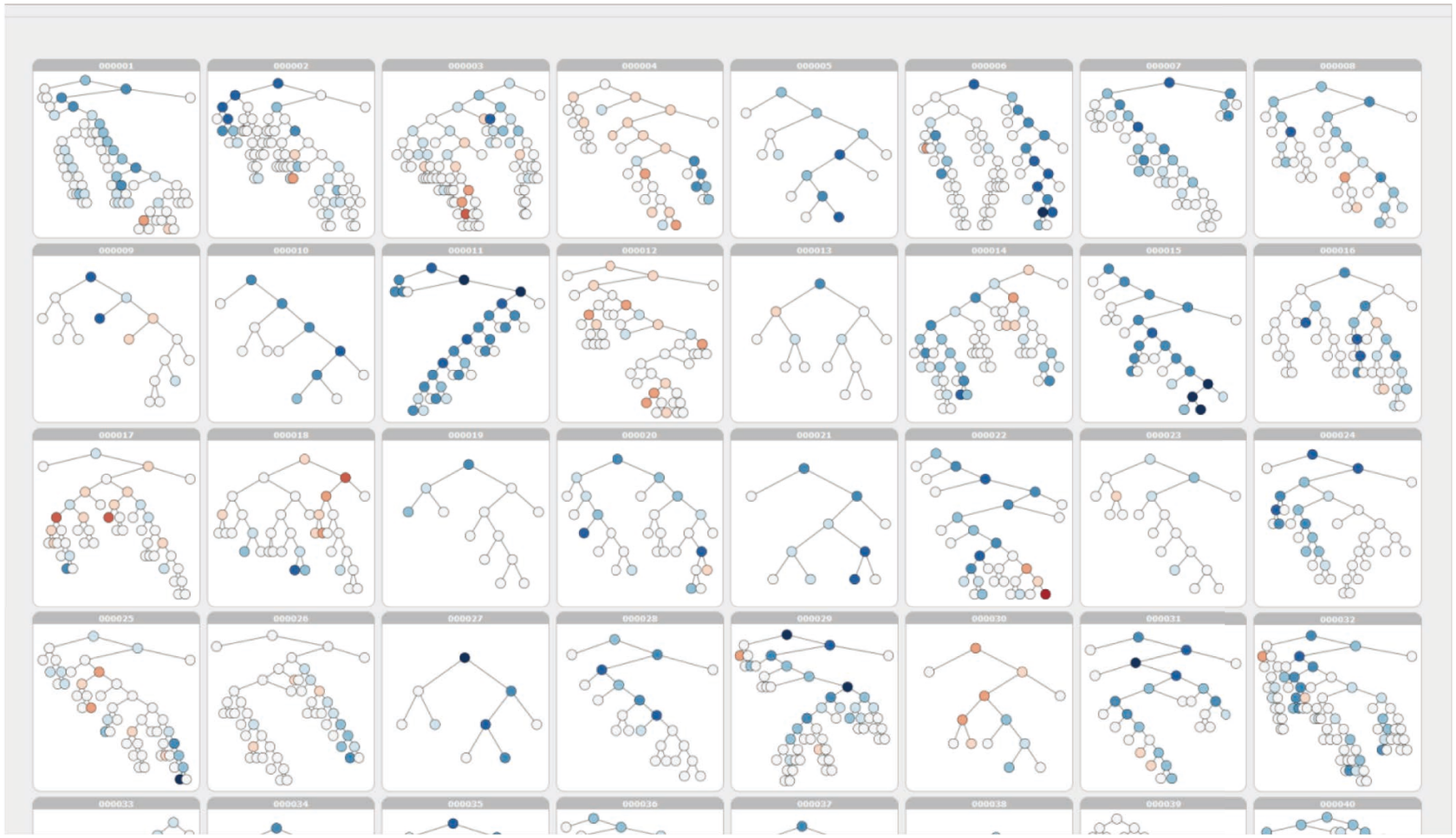
- Is the tone of a piece of text positive, negative, or neutral?

> I can't remember the last time I saw a movie that contained as many genres as 'Parasite'. The movie starts out almost like an 'Ocean's Eleven' heist film and then expands into a comedy, mystery, thriller, drama, romance, crime and even horror film. It really did have everything and it was strikingly good at all of them too.

Source: IMDB

Sentiment Analysis: Stanford Sentiment Treebank

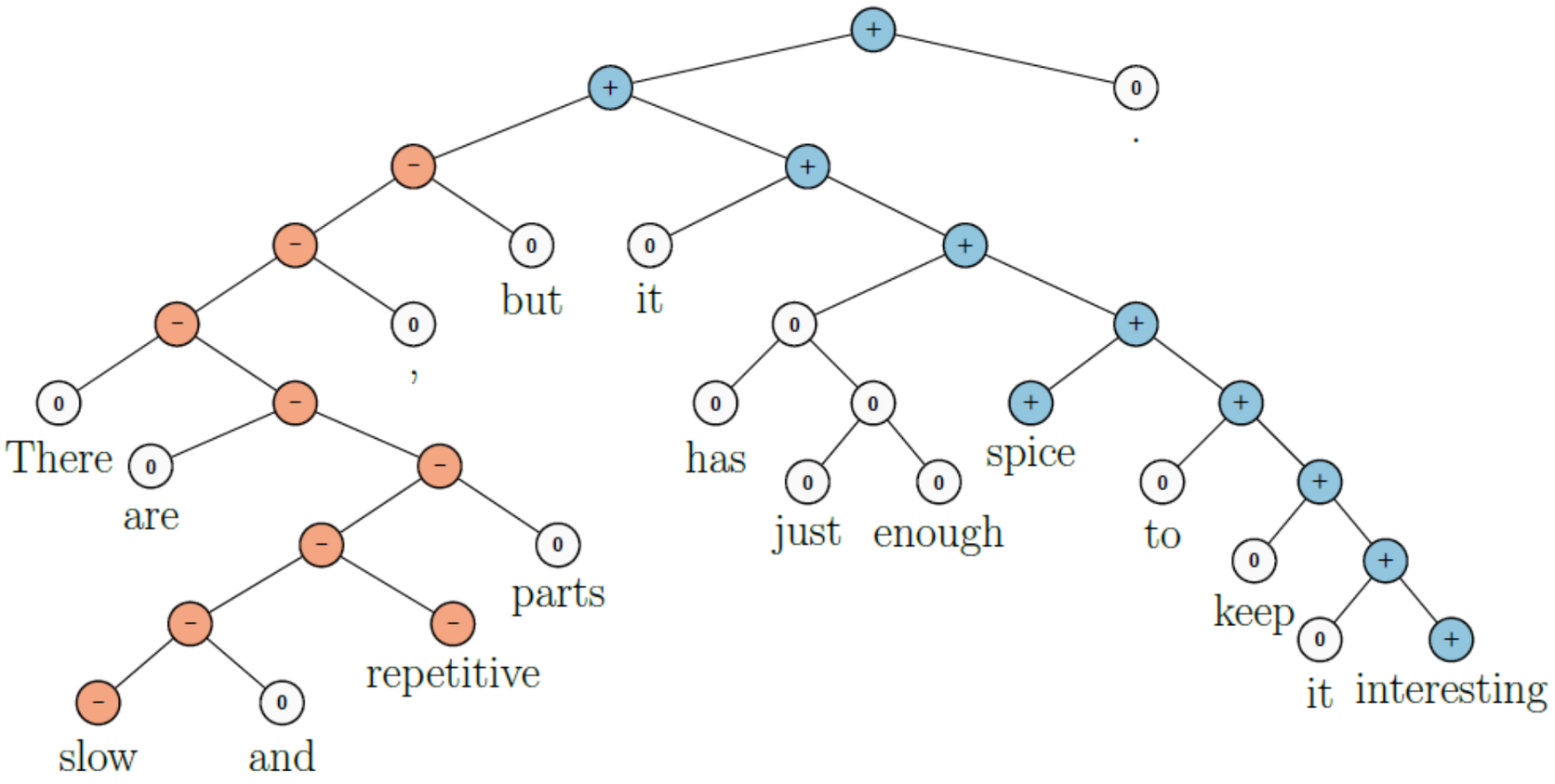
- 215,154 phrases labeled in 11,855 sentences
- Can actually train and test compositions



https://nlp.stanford.edu/sentiment/index.html

Source: Stanford 224n

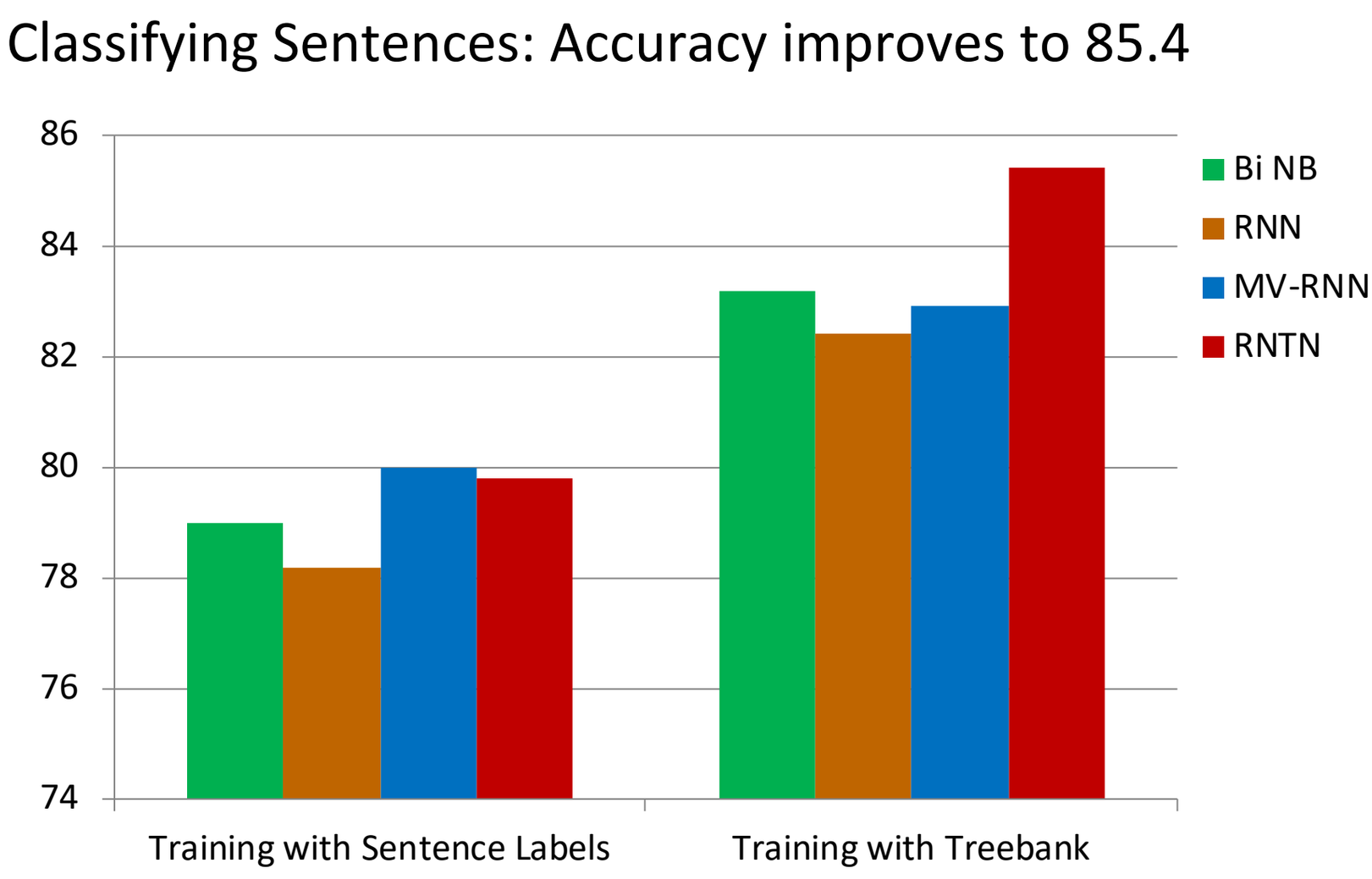
Sentiment Analysis: Stanford Sentiment Treebank



https://nlp.stanford.edu/sentiment/index.html

Source: Stanford 224n

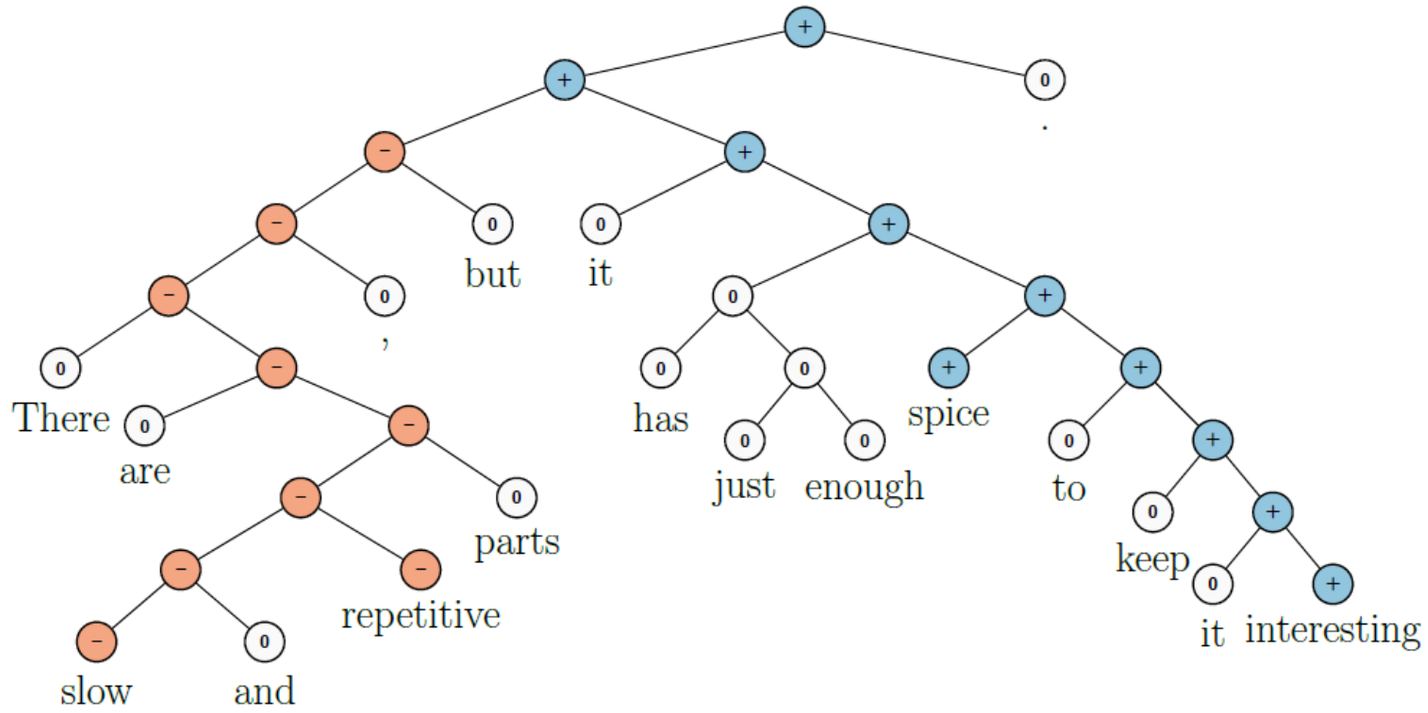
Sentiment Analysis



Positive/Negative

Source: Stanford 224n

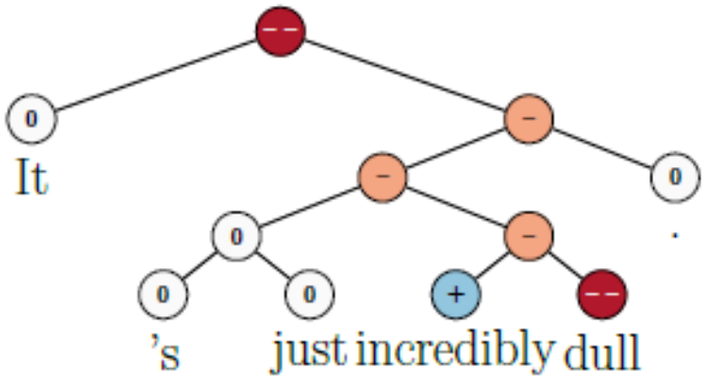
Sentiment Analysis

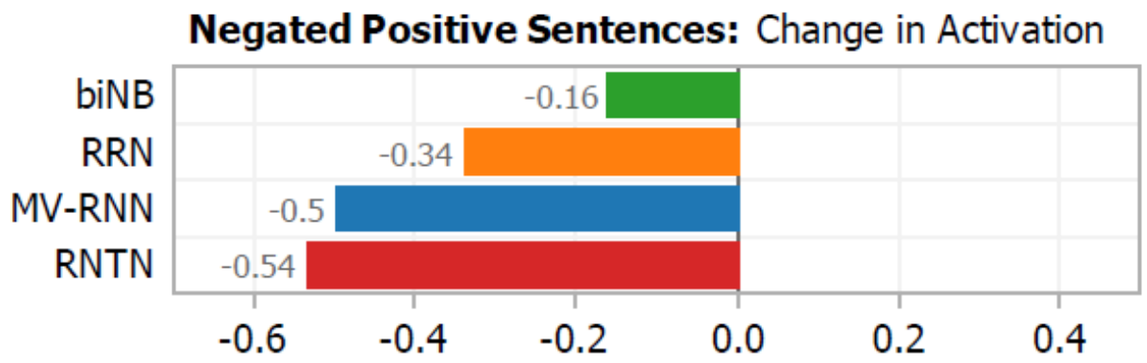


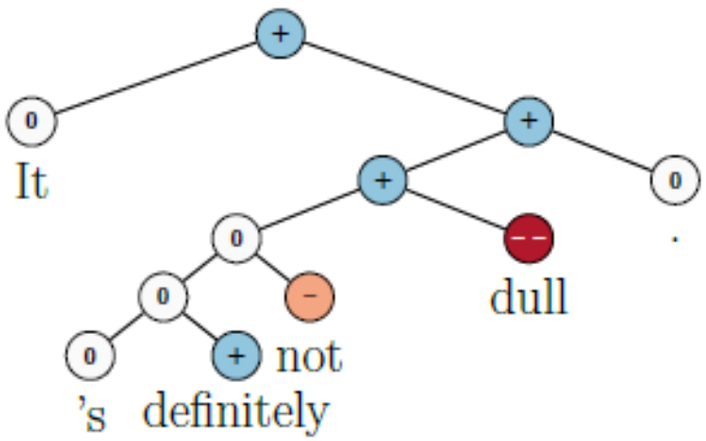
- RNTN can capture constructions like X but Y
- RNTN accuracy of 72%, compared to MV-RNN (65%), bi-word NB (58%) and RNN (54%)

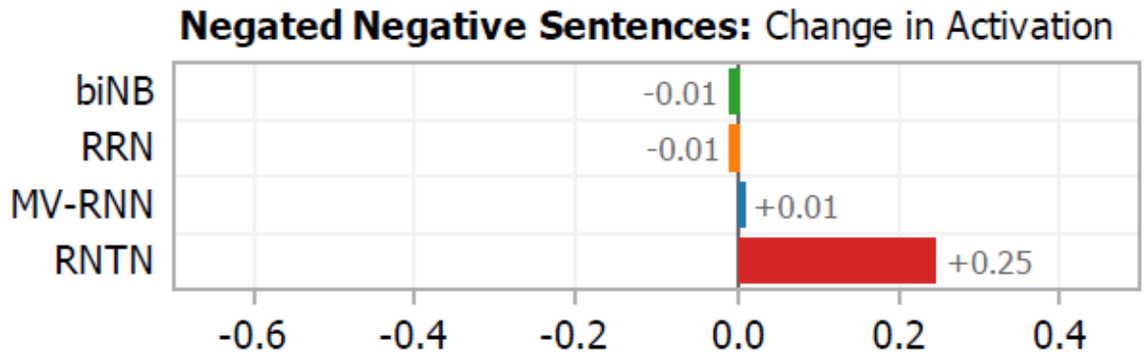
Source: Stanford 224n

Sentiment Analysis









When negating negatives, positive activation should increase!

Source: Stanford 224n

Version 5: Tree LSTM

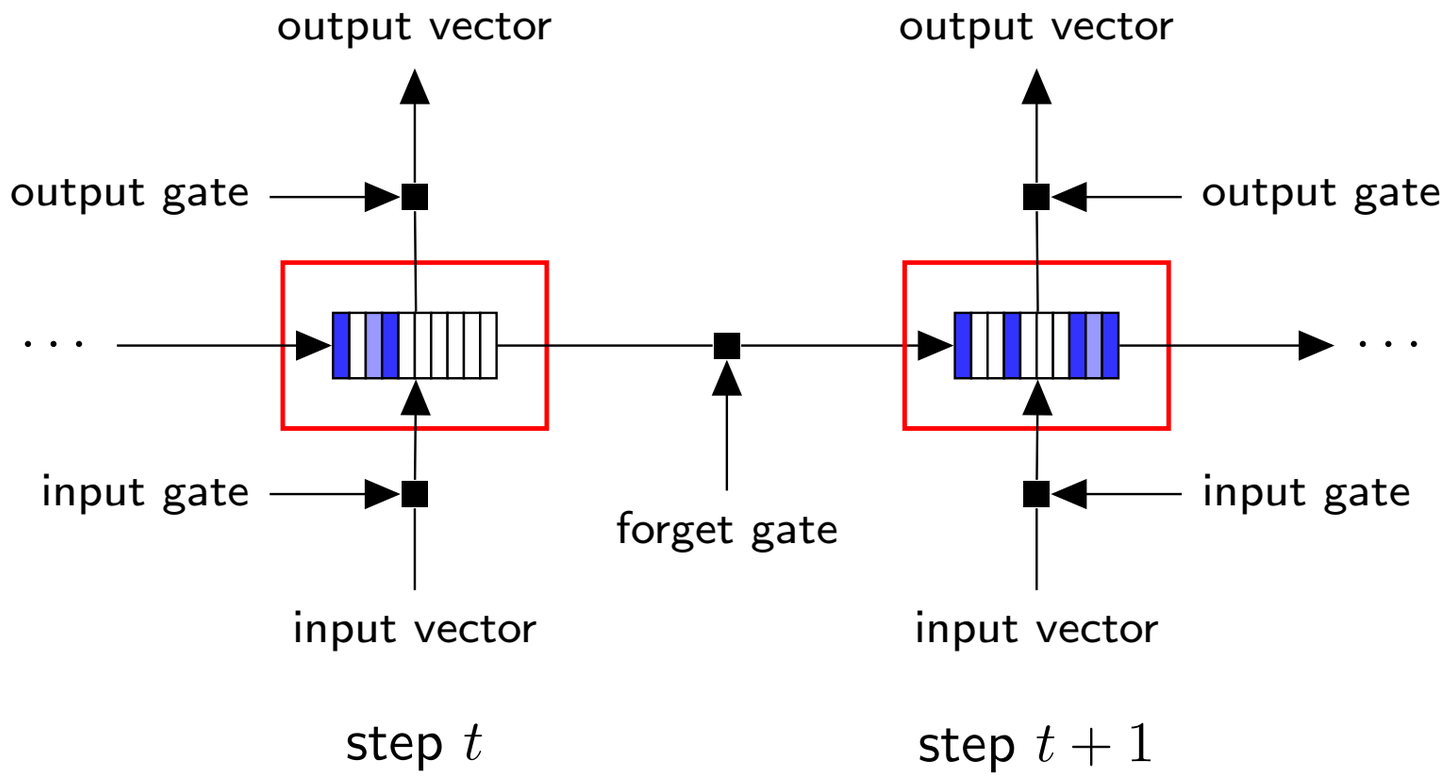
1. Generalization of sequential LSTM to tree structures.
2. Similar to recursive networks, it uses the **tree structure** to recursively **encode** node representations.
3. Generalizes to any number of children (not just binary).

- Goal:

To represent the meaning of a sentence as a point in a (high-dimensional, continuous) vector space in a way that accurately handles semantic composition and sentence meaning

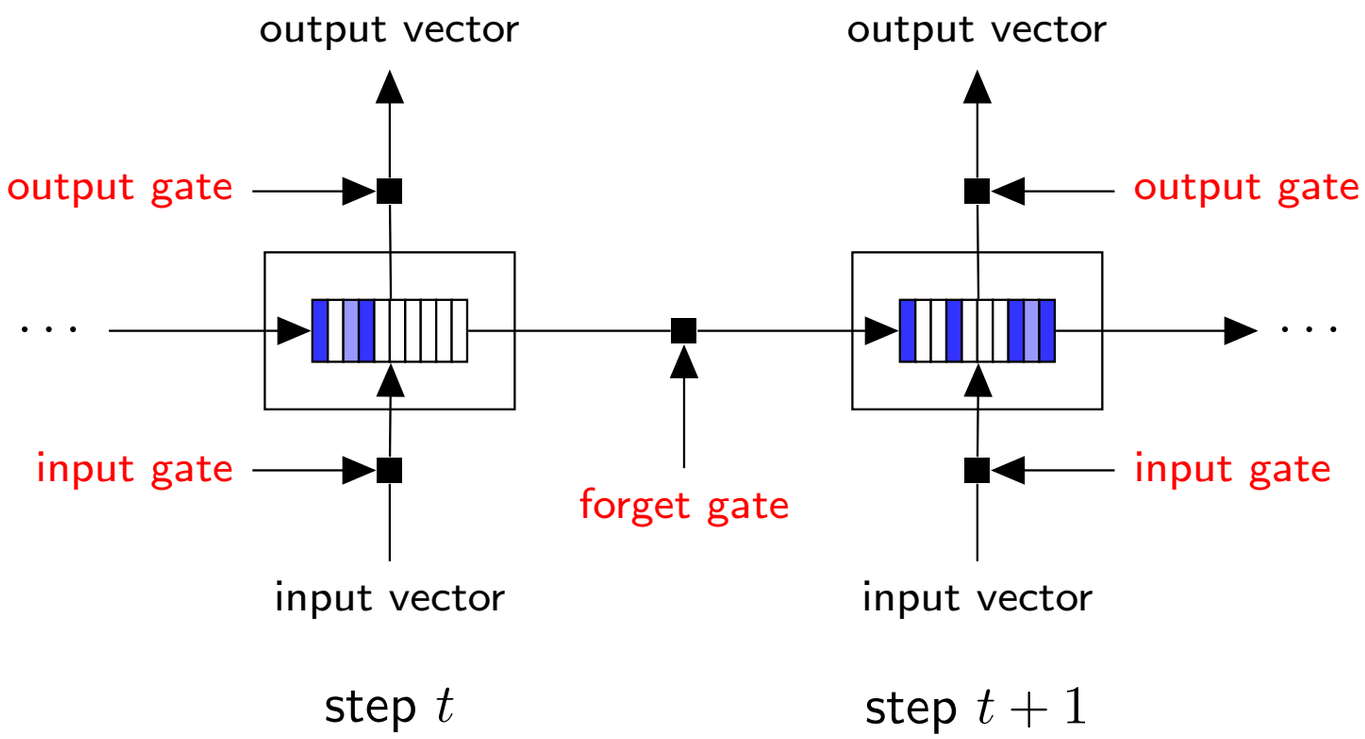
Sequential LSTM

ϕ is the composition function to be learned from data.



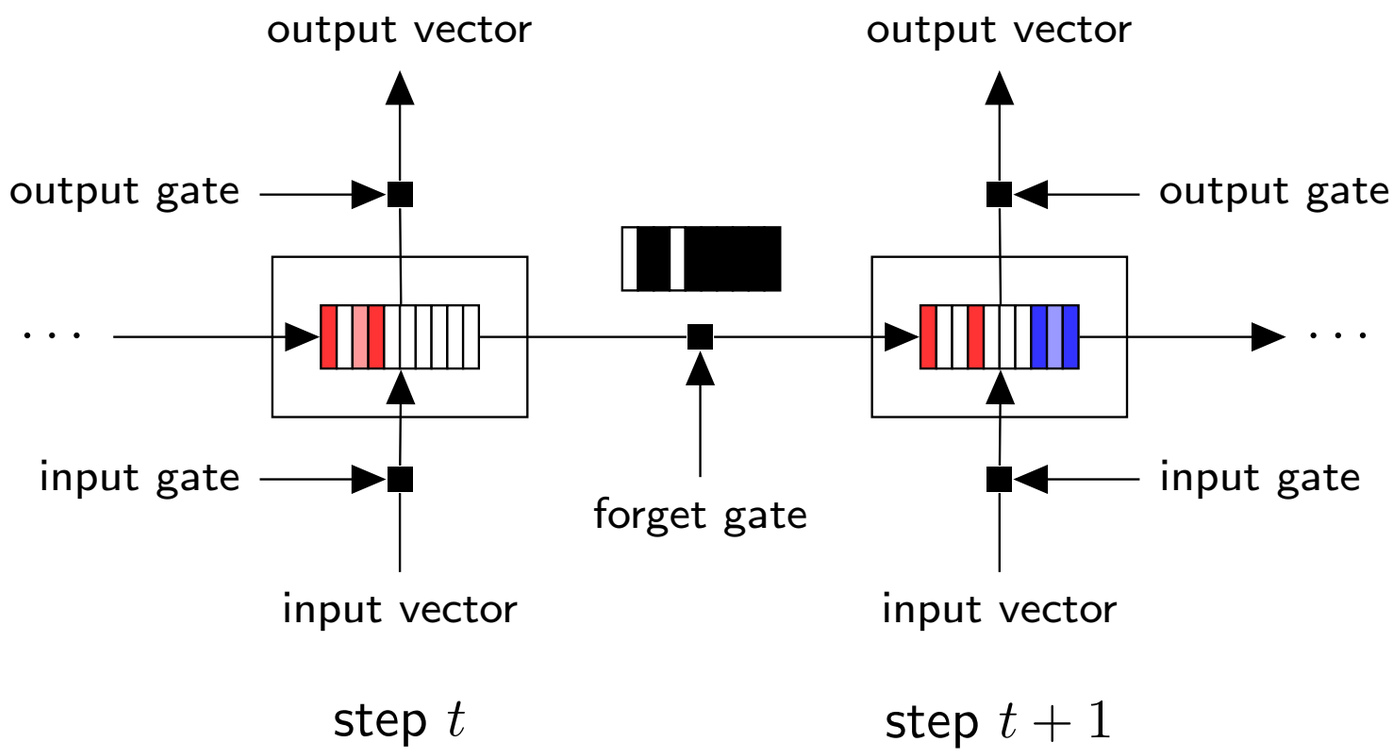
1. LSTM is a particular parameterization of ϕ.
2. Memory cell: a vector representing the inputs seen so far. Its state can be preserved over many time steps.

Sequential LSTM



1. input/output/forget gates are vectors in $[0,1]^d$, used as **soft mask** (i.e., by element-wise multiplication) for selective memory read/write/information-propagation.
2. Each gate is **parameterized and conditioned** on the current input $\mathbf{x}_t$ and previous hidden state $\mathbf{h}_t$ (*output vector* in Fig).

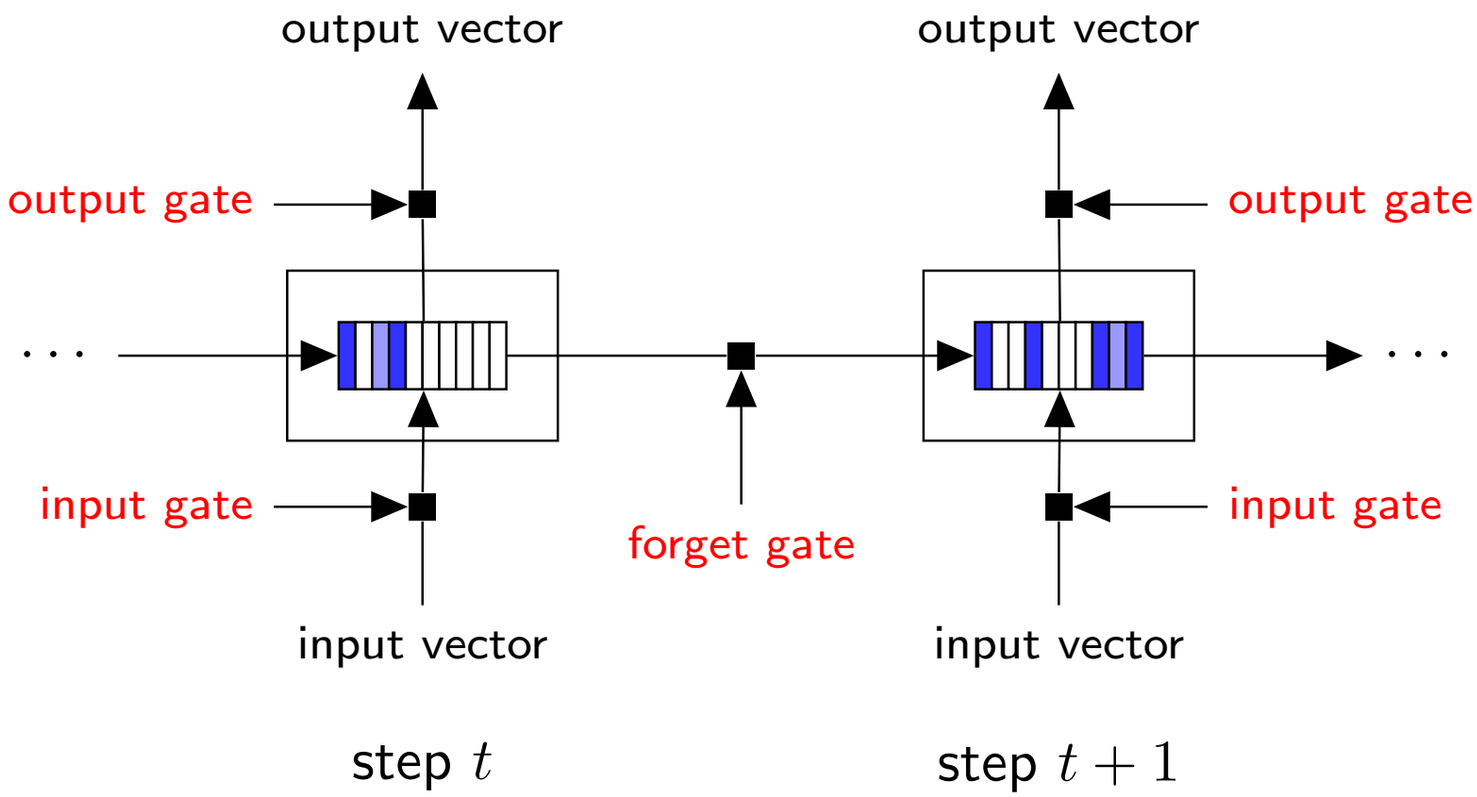
Sequential LSTM



Starting at state $t-1$:

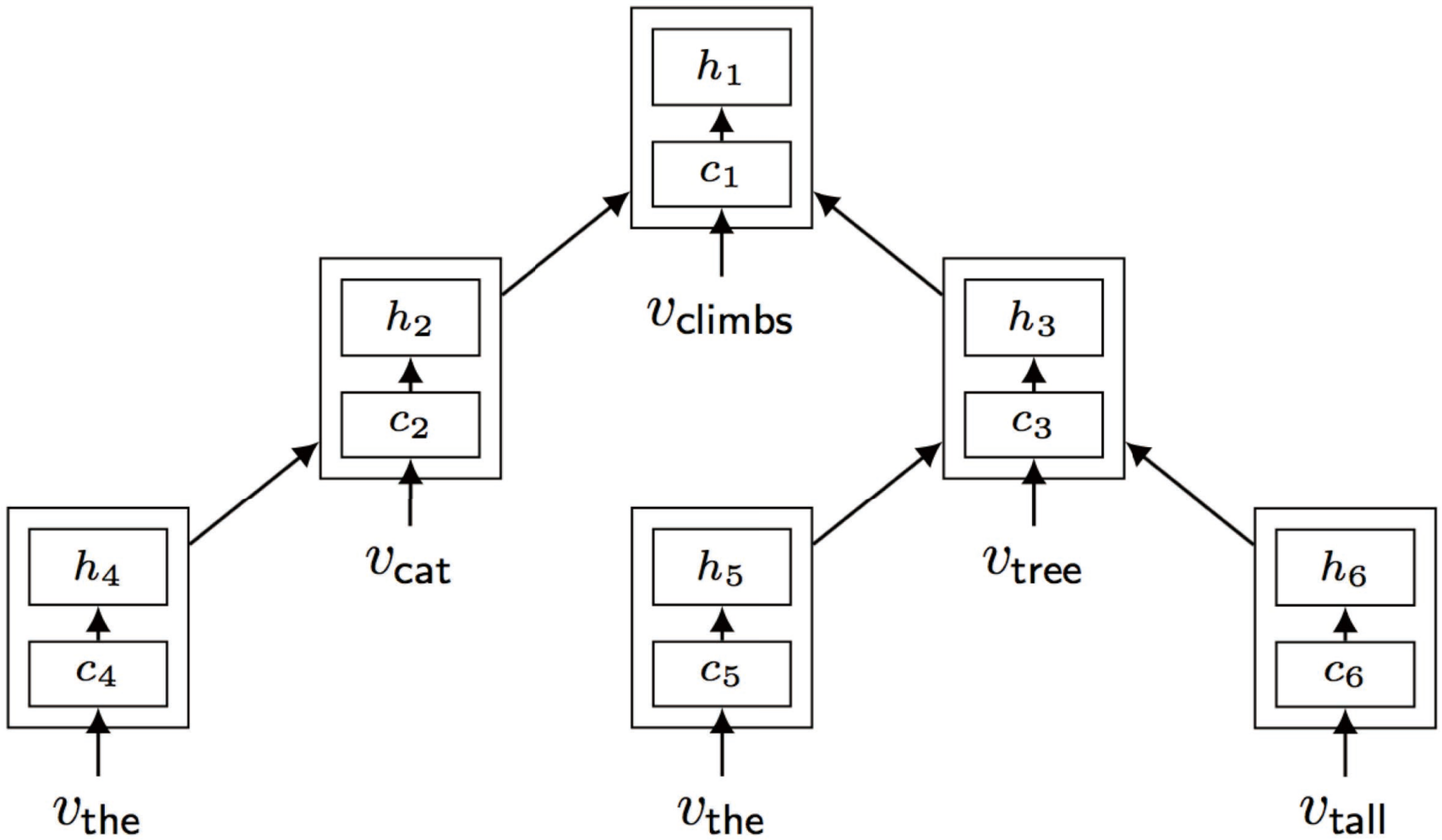
1. Predict gates $(\mathbf{i}_t, \mathbf{f}_t, \mathbf{o}_t)$ and intermediate state $(\bar{\mathbf{c}}_t)$ from $\mathbf{x}_t$ and $\mathbf{h}_{t-1}$.
2. Update cell state: $\mathbf{c}_t = \mathbf{i}_t \otimes \bar{\mathbf{c}}_t + \mathbf{f}_t \otimes \mathbf{c}_{t-1}$.
3. Compute hidden state (output vector): $\mathbf{h}_t = \mathbf{o}_t \otimes \mathbf{c}_t$

Generalising LSTM

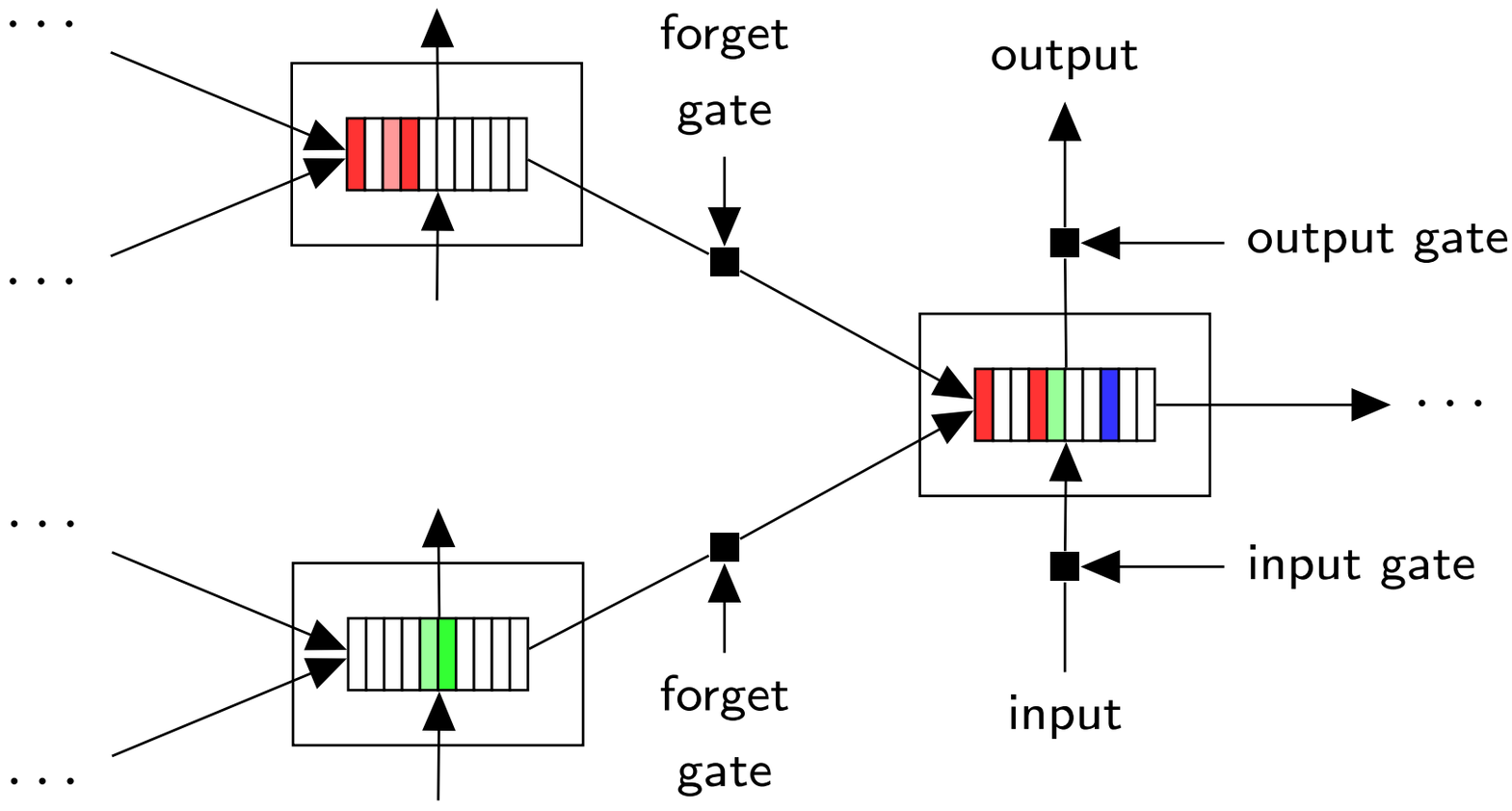


1. Sequential LSTM receives only one child at a time.
2. Generalize it to receive **multiple children.**

Tree LSTM

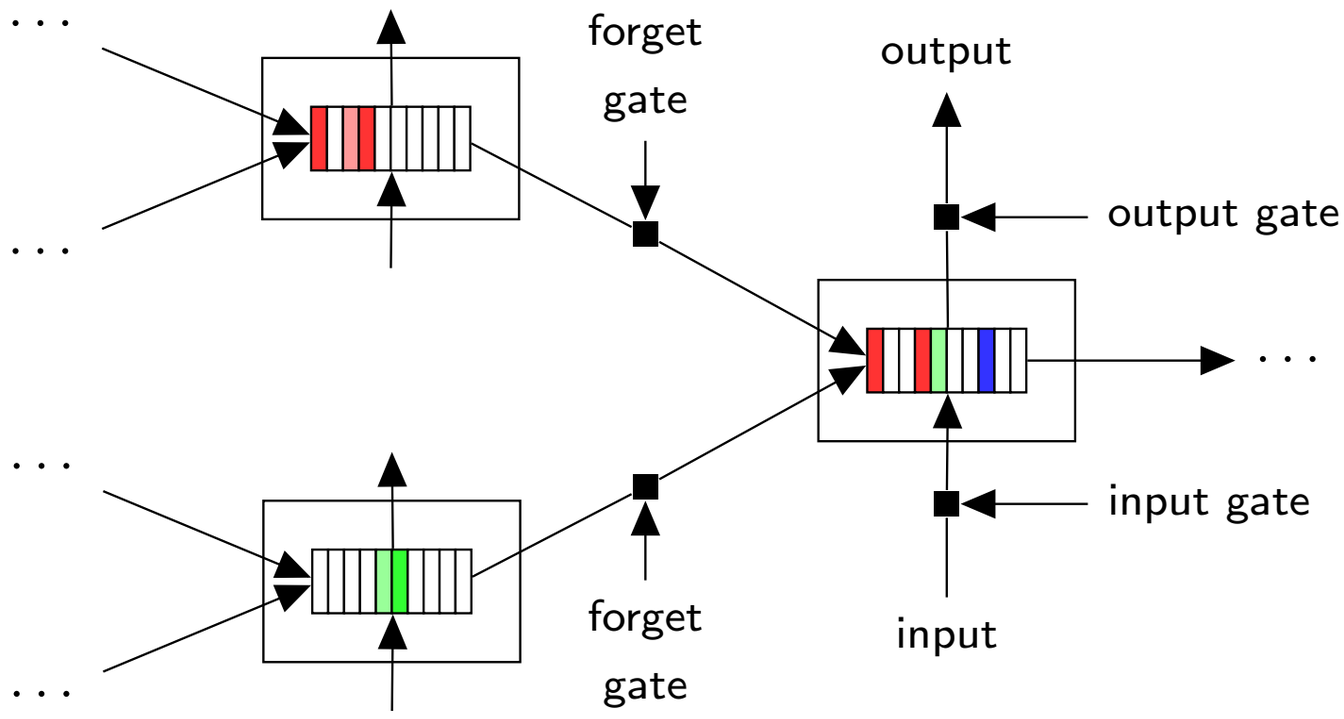

h_1
c_1
v_{climbs}
h_2
c_2
v_{cat}
h_3
c_3
v_{tree}
h_4
c_4
v_{the}
h_5
c_5
v_{the}
h_6
c_6
v_{tall}

Tree LSTM



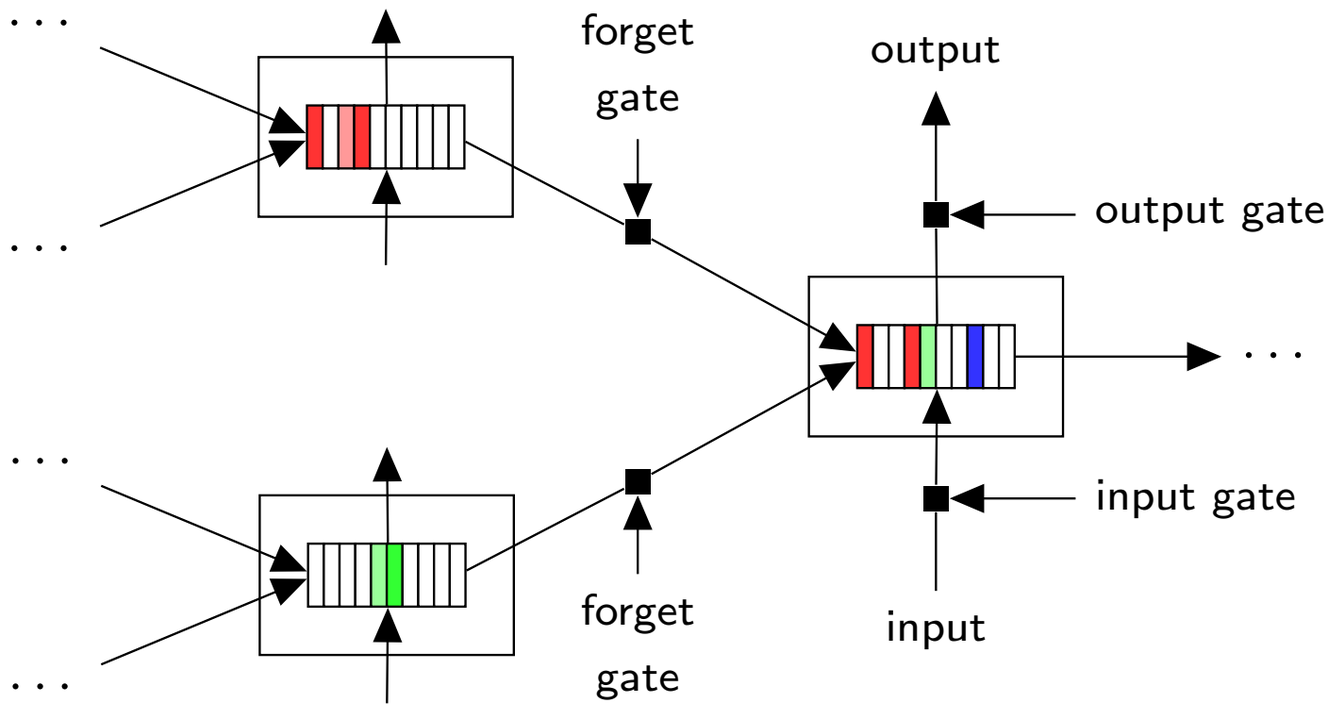
1. Allows arbitrary **branching factor**.
2. Each branch has a **separate forget gate**.

Tree-Sum LSTM



1. **Sum up** all child vectors for computing gates and cell states.
2. Forget gate parameters are **shared across branches.**
3. Supports trees with variable branch numbers.

N-ary Tree LSTM



1. Each branch uses its own parameters for computing gates and cells.
2. Supports trees with a **max branching factors**.

Evaluation (Sentiment Analysis)

Method	5-class	Binary	
RNTN (Socher et al., 2013)	45.7	85.4	
Paragraph-Vec (Le & Mikolov, 2014)	48.7	87.8	
Convolutional NN (Kim 2014)	47.4	**88.1**	
Epic (Hall et al., 2014)	49.6	–	
DRNN (Irsoy & Cardie, 2014)	49.8	86.6	
LSTM	46.4	84.9	⋆
Bidirectional LSTM	49.1	87.5	⋆
Constituency Tree-LSTM	**51.0**	88.0	⋆

Evaluation (Semantic Relatedness)

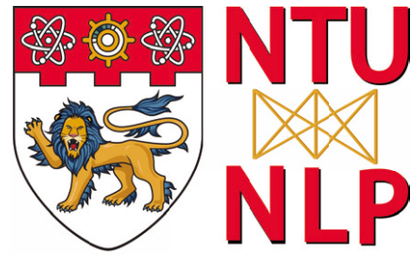
"the snowboarder is leaping over white snow" $\overset{?}{\sim}$ "a person who is practicing snowboarding jumps into the air"

- **Task:** Predict the semantic relatedness of sentence pairs
- **Dataset:** SICK from SemEval 2014, Task 1
- **Supervision:** Human-annotated relatedness scores $y \in [1, 5]$.
- **Model:** Sentence representation with Tree-LSTM on dependency parses. Similarity predicted by NN regressor given representations at root nodes

Evaluation (Semantic Relatedness)

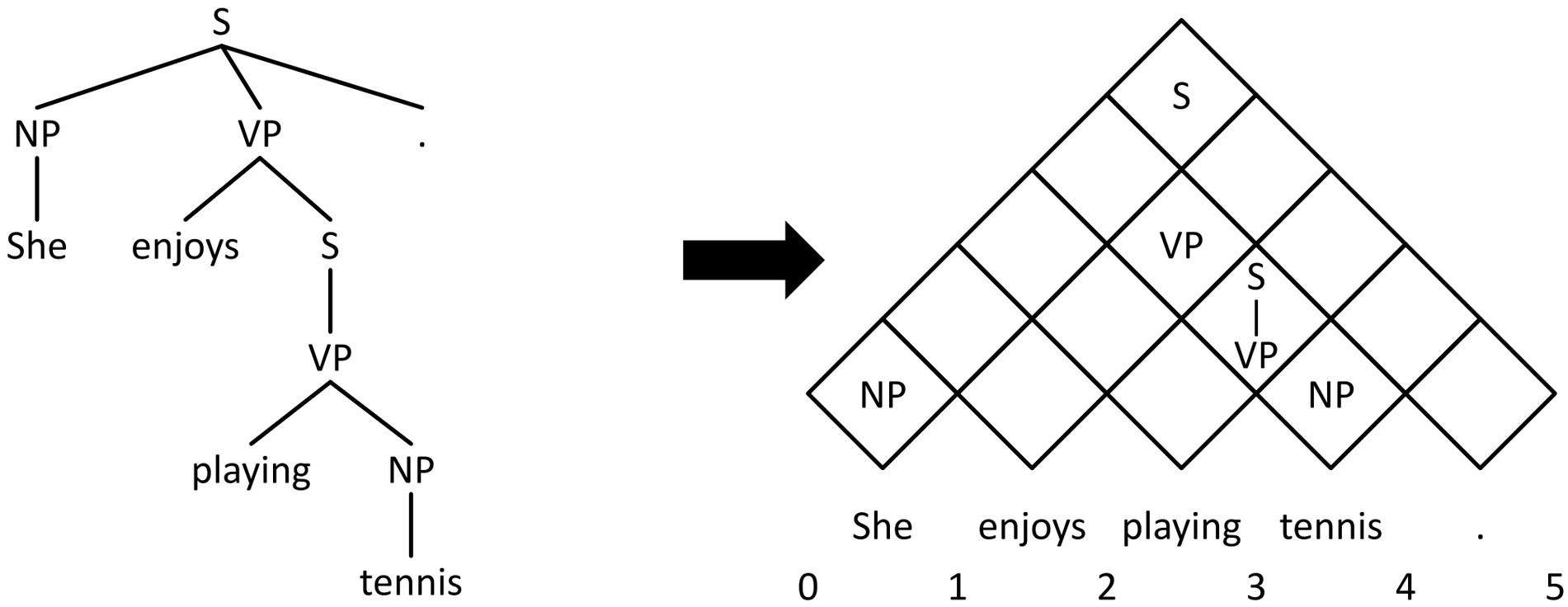
Method	Pearson's r	
Word vector average	0.758	
Meaning Factory (Bjerva et al., 2014)	0.827	
ECNU (Zhao et al., 2014)	0.841	
LSTM	0.853	$\star$
Bidirectional LSTM	0.857	$\star$
Dependency Tree-LSTM	**0.868**	$\star$

Today's Plan



- Syntactic structures
  - Constituency & Dependency structures
- Need for syntactic structures
- Are languages hierarchical?
- Parsing with Recursive Neural Networks
- Backpropagation through tree
- Semantic composition with Tree RNNs
- Recent Parsers

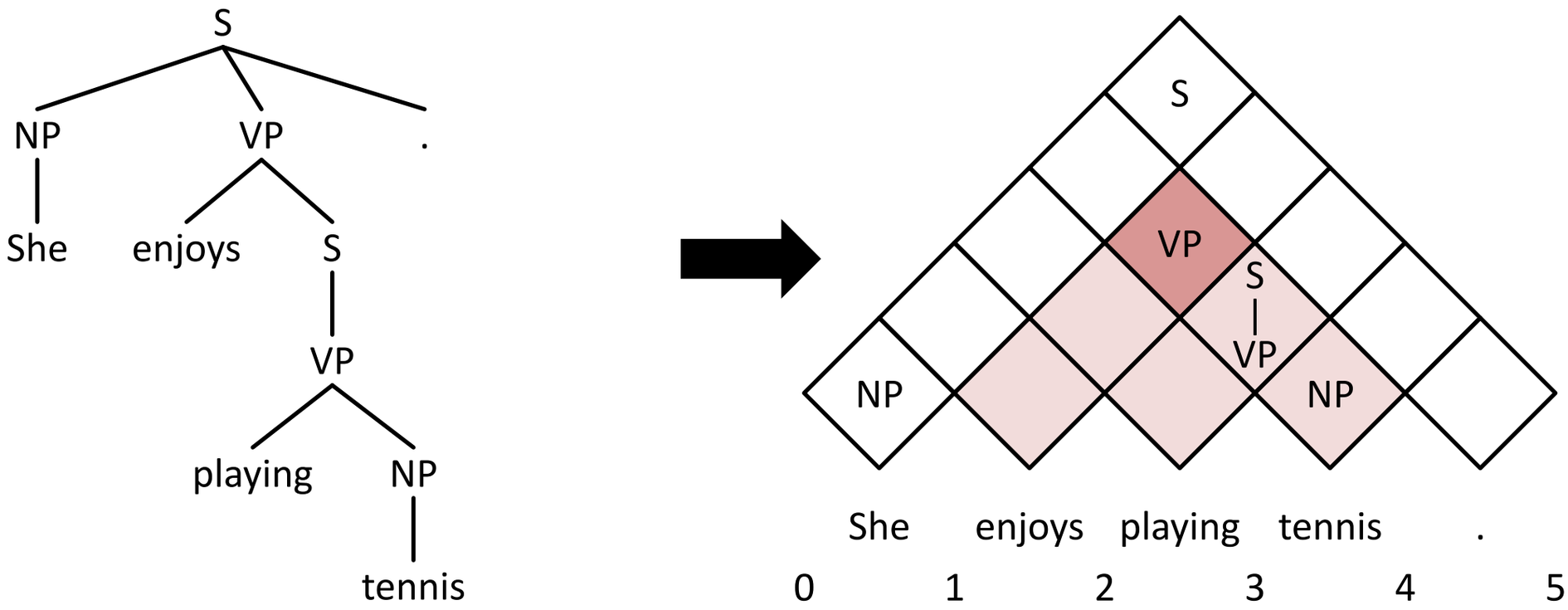
Chart Parsing



Constituency parses can be represented by a chart

Source: Stern et al, 2017

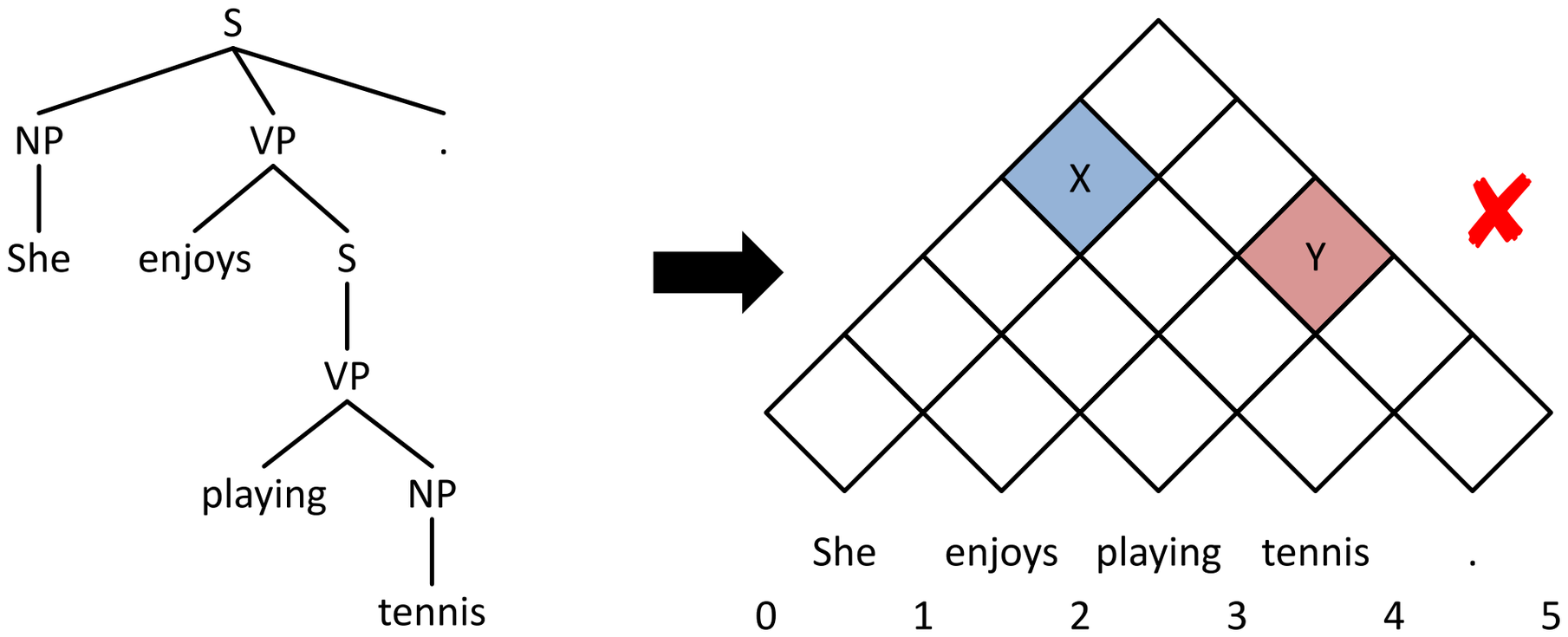
Chart Parsing



Constituency parses can be represented by a chart

Source: Stern et al, 2017

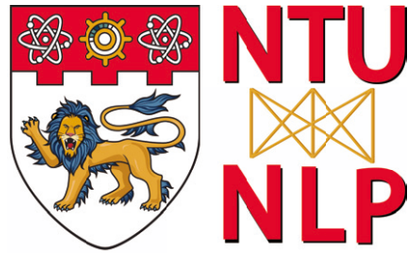
Chart Parsing



Needs tree constraints

Source: Stern et al, 2017

Use of CFG



S(decided) → NP(workers) VP(decided) [Collins (1999)]

S → NP^S VP^S [Klein and Manning (2003)]

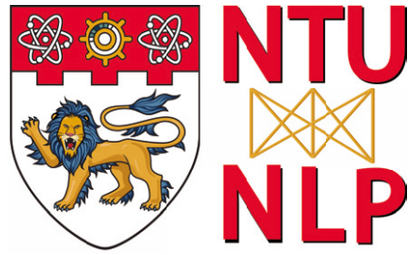
S → NP VP [Hall et al. (2014)]

S Modern Parsers

Modern Parsers don't use any CFGs; uses only labels

Source: Stern et al, 2017

Scoring in Parsing



score(S → NP^S VP^S) [Klein and Manning (2003)]

score(*i*, *k*, *j*, S → NP VP) [Hall et al. (2014)]

score(*i*, *j*, S) and $\text{score}_{\text{action}}(i, k, j)$ [Cross and Huang (2016)]

Shift-Reduce

score(*i*, *j*, S)

Modern Parsers

Source: Stern et al, 2017

Decoding in Parsing

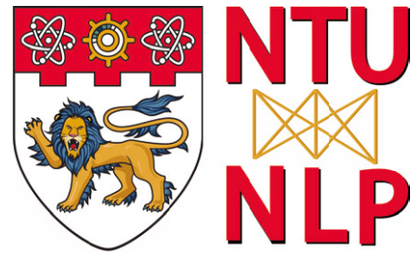


Chart-based

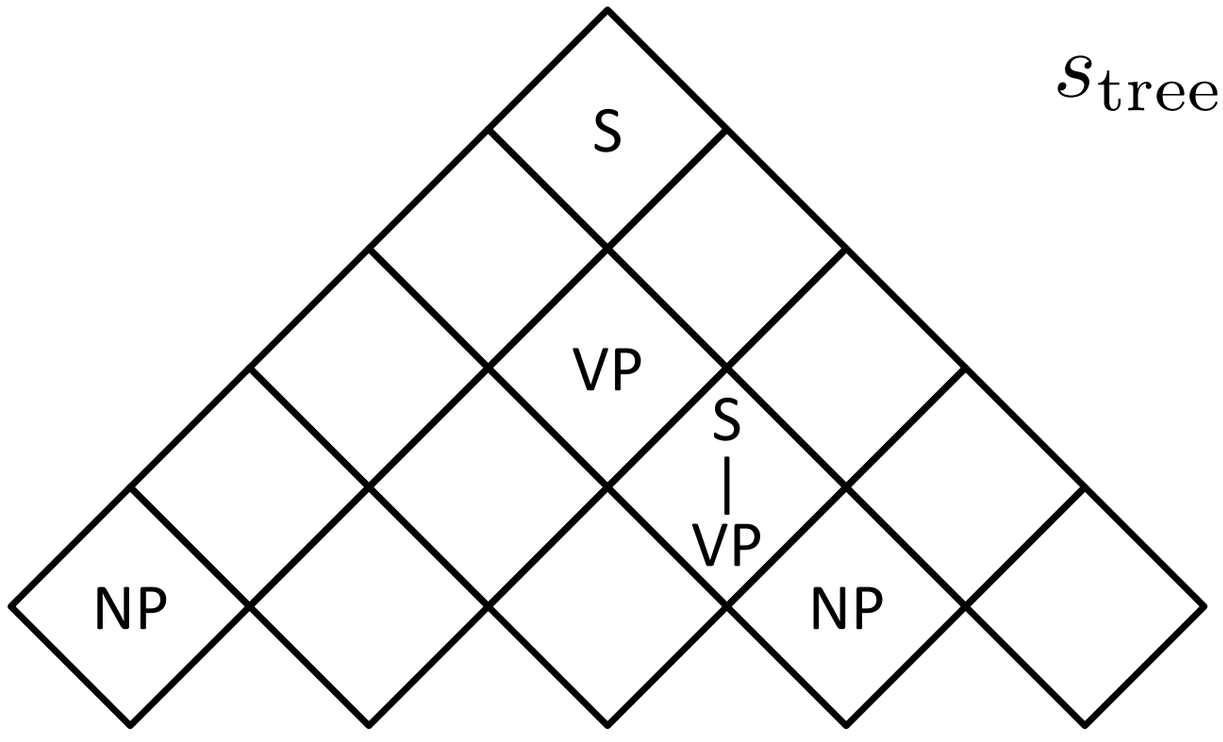
Globally optimal, $O(n^3)$ time complexity

Transition-based

Greedy, $O(n)$ or $O(n^2)$ time complexity

Source: Stern et al, 2017

Scoring in Modern Parsers

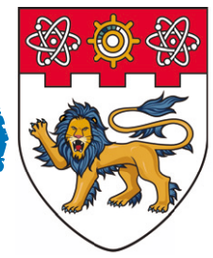


$$s_{\text{tree}}(T) = \sum_{(\ell,(i,j))\in T} s(i,j,\ell)$$

$$\begin{aligned}
&= s(0,5,\text{S}) \\
&\quad + s(0,1,\text{NP}) \\
&\quad + s(1,4,\text{VP}) \\
&\quad + s(2,4,\text{S–VP}) \\
&\quad + s(3,4,\text{NP})
\end{aligned}$$

Source: Stern et al, 2017

Dynamic Programming (CKY)



NTU NLP

Base case:

Pick best label

$$s_{\text{best}}(i, i+1) = \max_{\ell}[s(i, i+1, \ell)]$$

General case:

Pick best label

Bottom Up

$$s_{\text{best}}(i, j) = \max_{\ell}[s(i, j, \ell)]$$

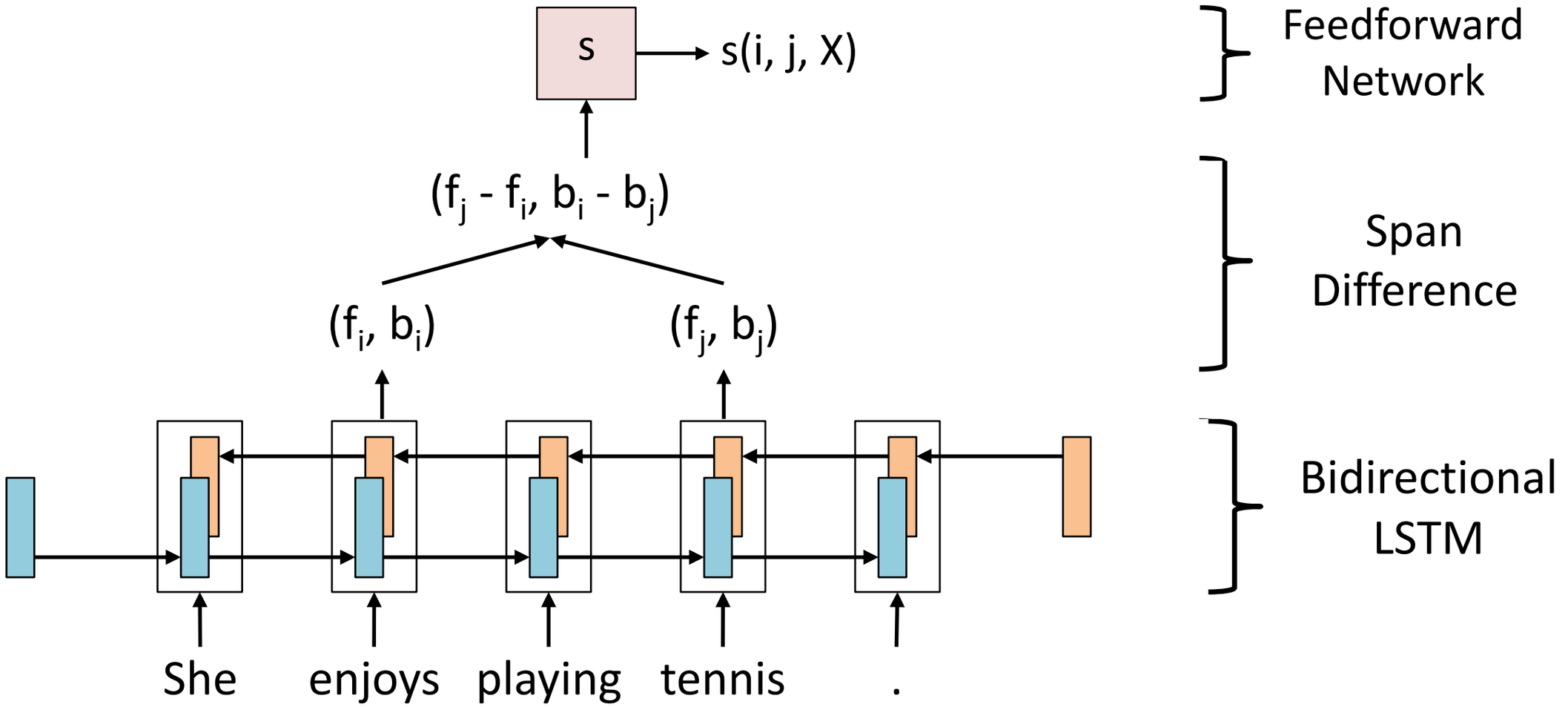
$$+ \max_{k}[s_{\text{best}}(i, k) + s_{\text{best}}(k, j)]$$

Pick best split point

Source: Stern et al, 2017

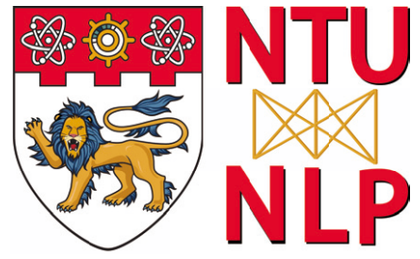
Scoring Functions

Using Sequential B-LSTM as the Sentence Encoder (Stern et al, 2017)



[Inspired by Cross and Huang (2016)]

Max-Margin Training



(Stern et al, 2017)

Want $s_{\text{tree}}(T^*) > s_{\text{tree}}(T)$ for all $T \neq T^*$

Require larger margin for higher loss:

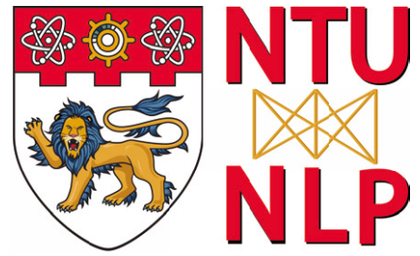
$$s_{\text{tree}}(T^*) \geq \Delta(T, T^*) + s_{\text{tree}}(T)$$

Use hinge penalty function:

$$\max\left(0, \Delta(\hat{T}, T^*) - s_{\text{tree}}(T^*) + s_{\text{tree}}(\hat{T})\right)$$

Source: Stern et al, 2017

Max-Margin Training



(Stern et al, 2017)

Use loss-augmented decoding during training:

$$\hat{T} = \max_{T} [\Delta(T, T^*) + s_{\text{tree}}(T)]$$

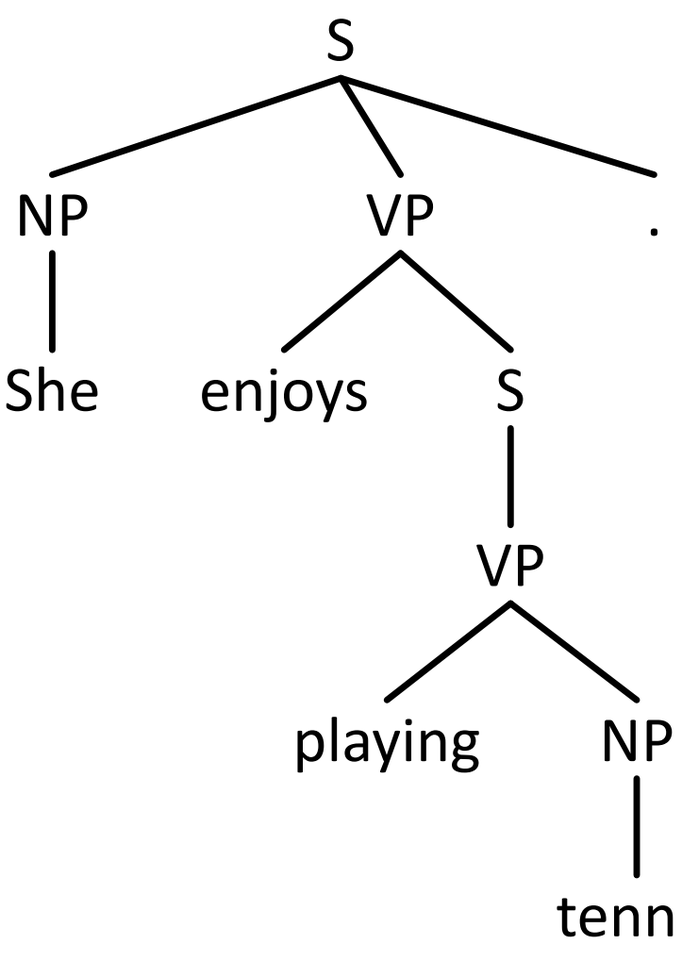
Loss-augmented decoding for Hamming loss:

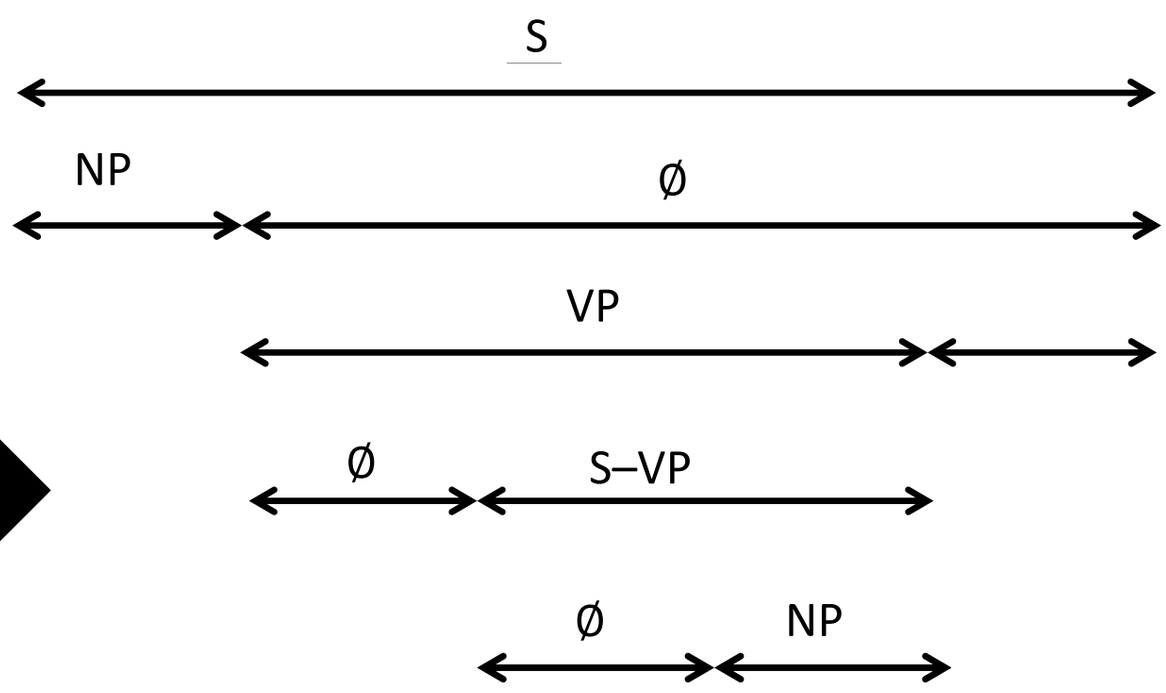
Replace $s(i, j, \ell)$ with $s(i, j, \ell) + \mathbf{1}(\ell \neq \ell^*_{ij})$

Use Dynamic Programming (CKY) to solve it

Source: Stern et al, 2017

Top-Down Greedy Parsing





Source: Stern et al, 2017

Top-Down Greedy Parsing

NTU NLP

$$\hat{\ell} = \underset{\ell}{\operatorname{argmax}}[s_{\text{label}}(i, j, \ell)]$$

$$\hat{k} = \underset{k}{\operatorname{argmax}}[s_{\text{span}}(i, k) + s_{\text{span}}(k, j)]$$

Training

Margin constraint for each decision:

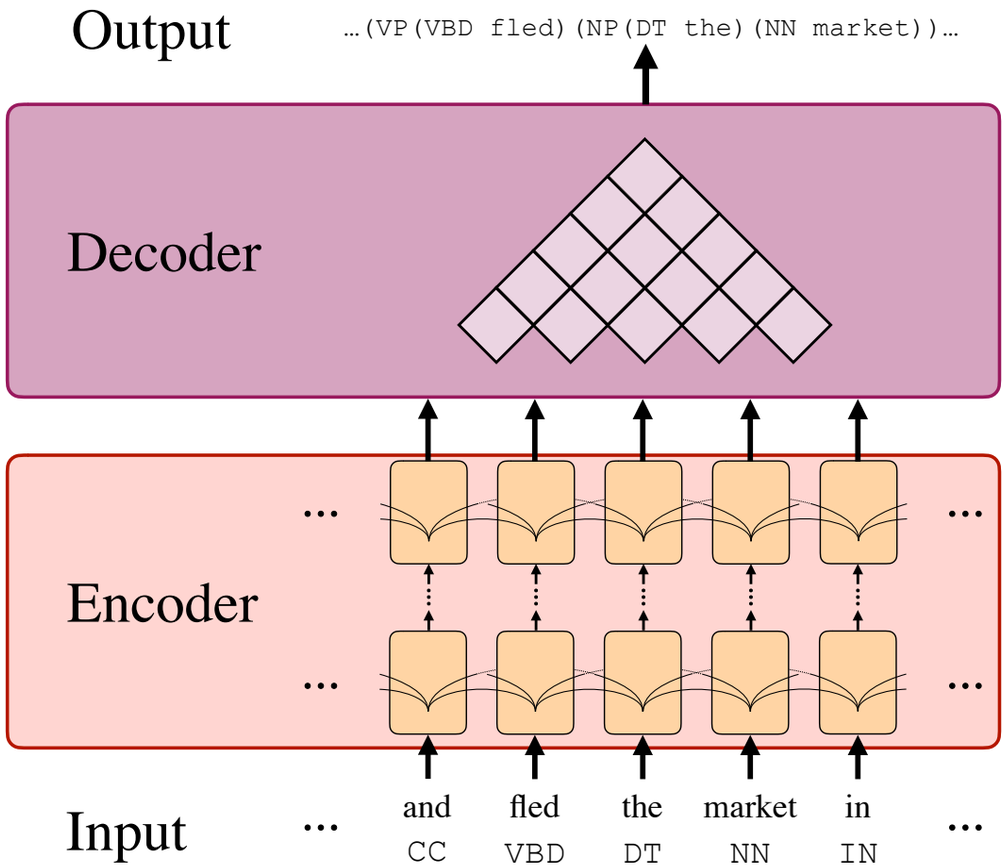
score(gold) ≥ 1 + score(other)

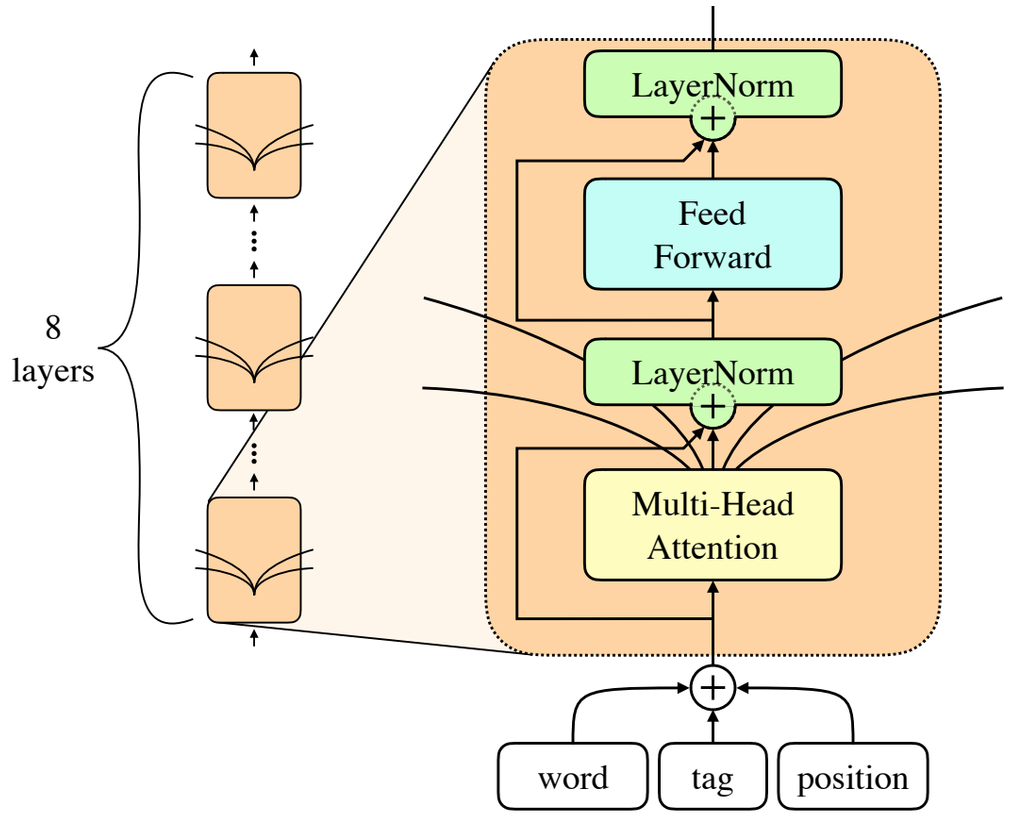
- Similar to CKY, but doesn't consider the best sub-trees underneath
- Neural encoder is powerful enough.

Source: Stern et al, 2017

Self-Attentive Encoder

Better Encoder



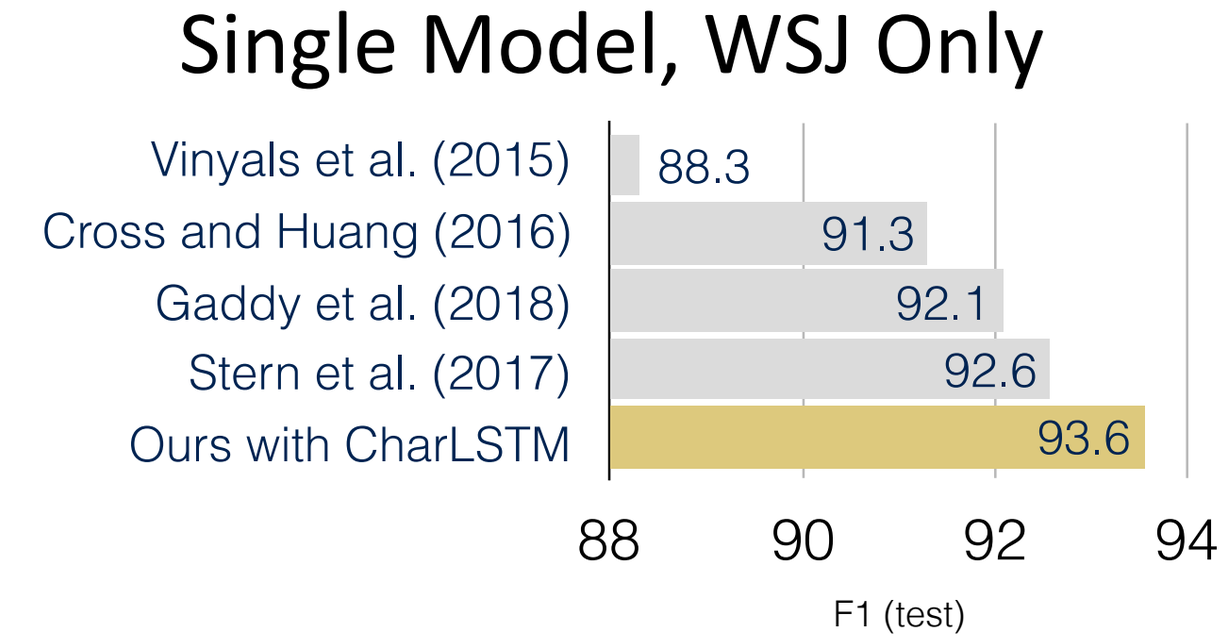


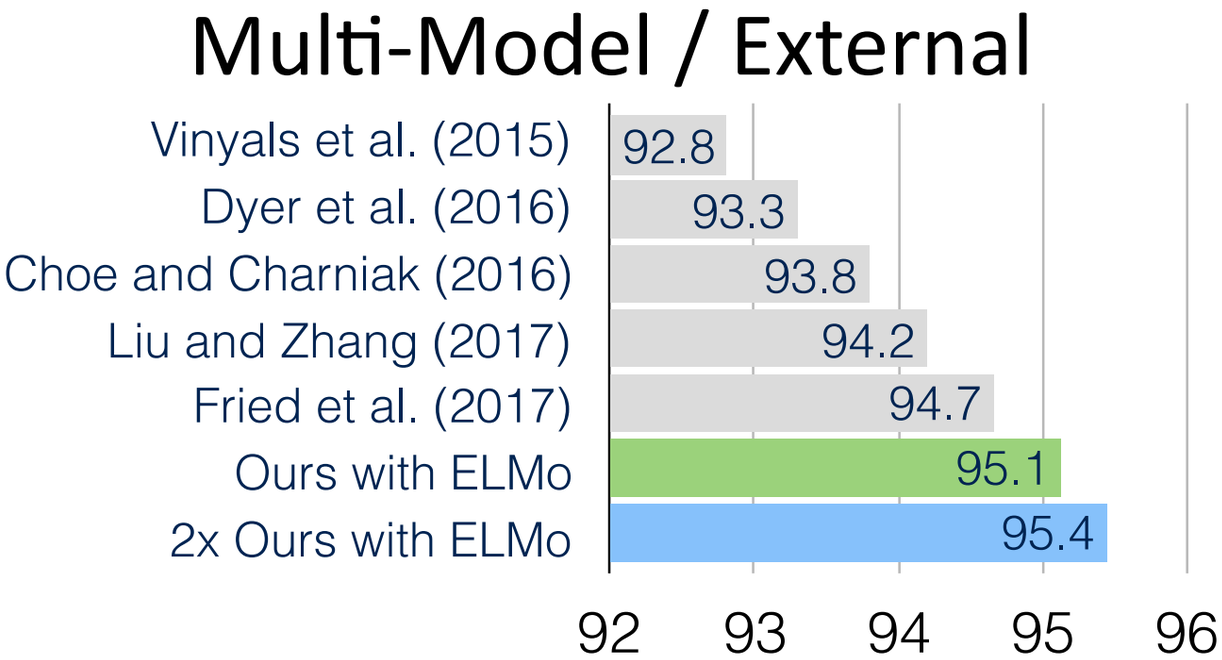
- Scoring and decoding are same as Stern et al, (2017)

Source: Kitaev & Klein, 2018

Results

English

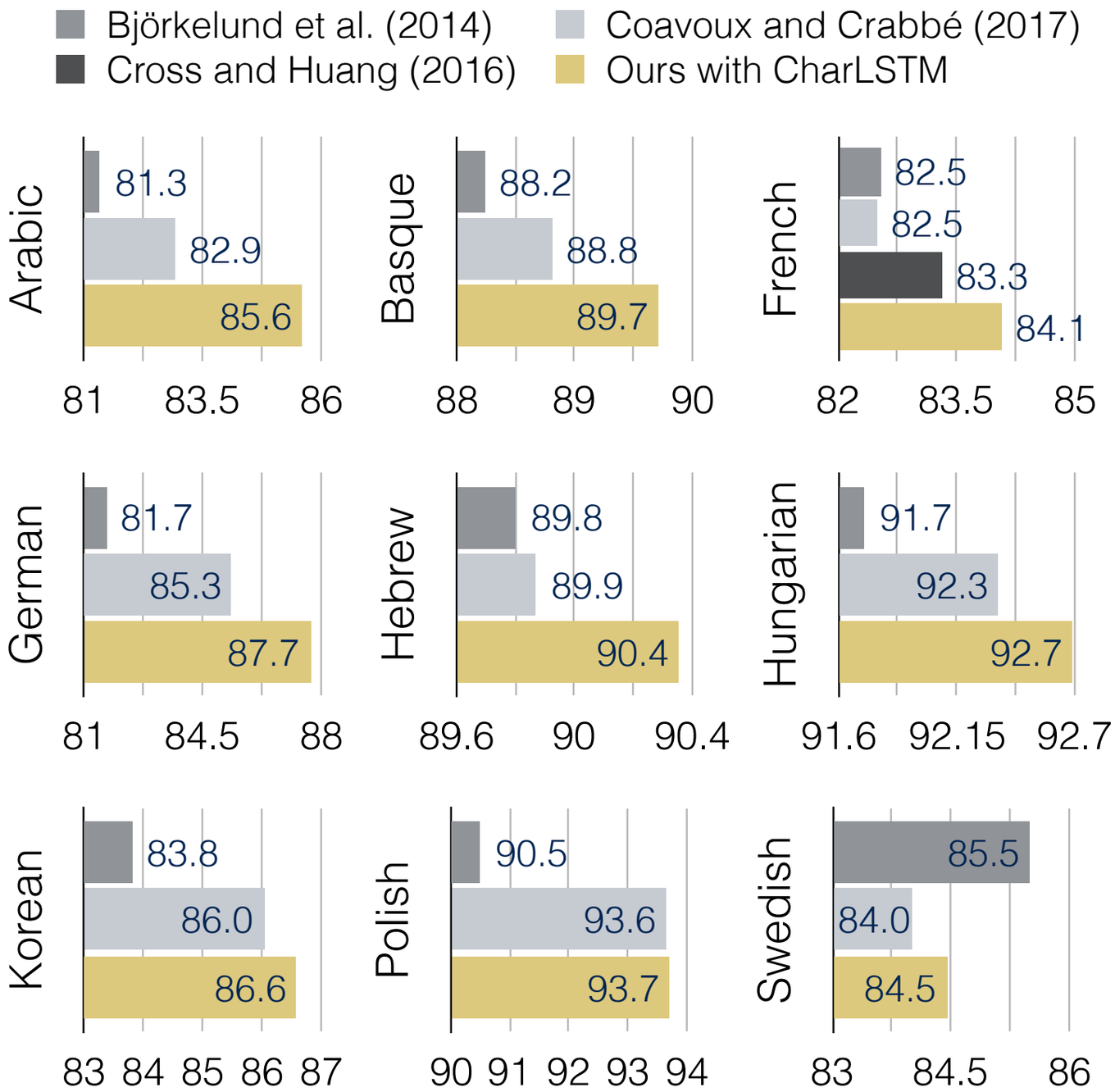




Source: Kitaev & Klein, 2018

Results

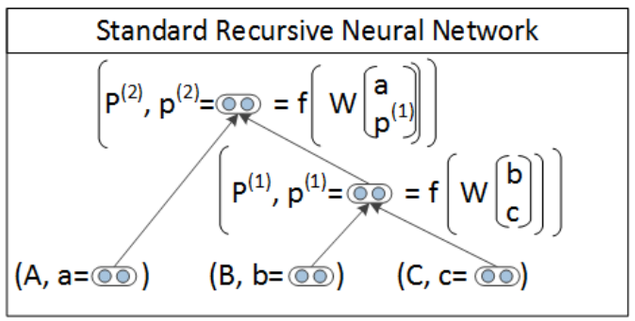
Multilingual (SPMRL)

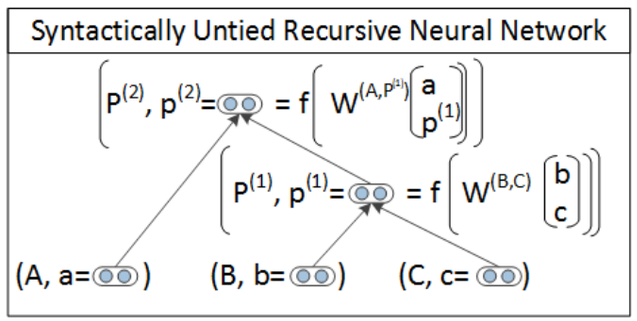


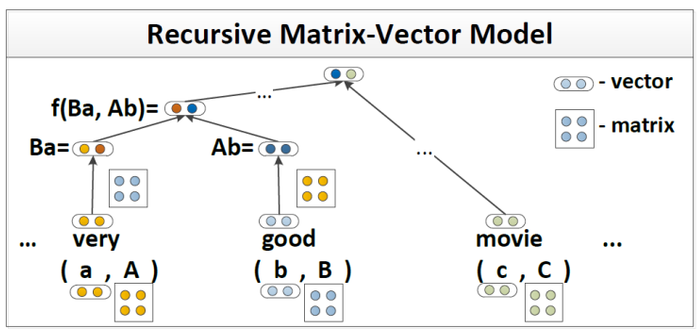
Source: Kitaev & Klein, 2018

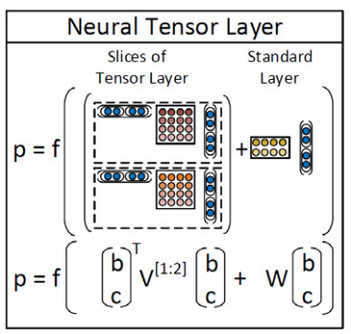
Summary

Tree-RNNs

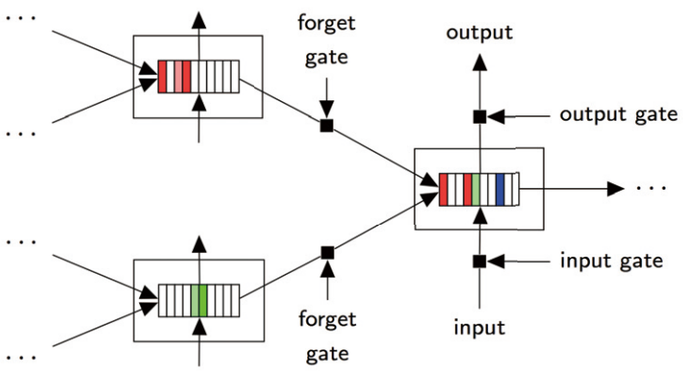








Tree-LSTM



Modern Parsers