CS224d: Deep NLP

Lecture 9:
Recursive Neural Networks

Richard Socher

richard@metamind.io

Overview

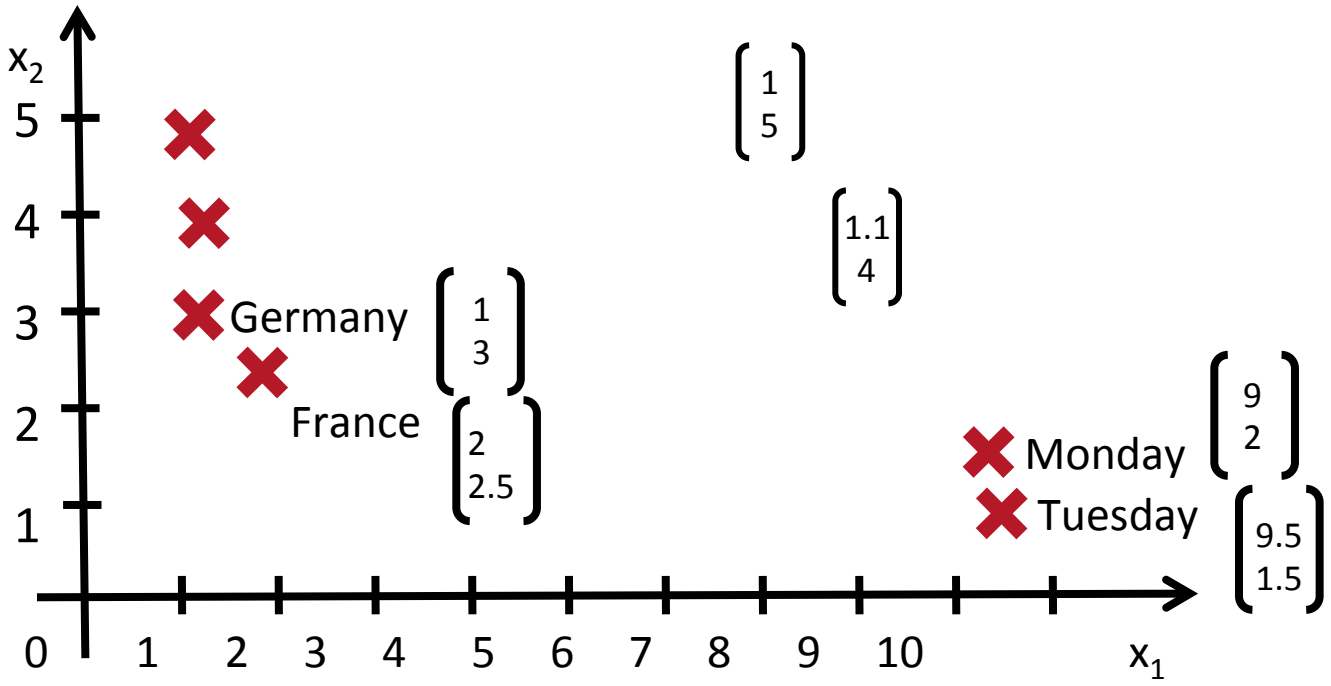
Recursive Neural Networks

- Motivation: Compositionality
- Structure prediction: Parsing
- Backpropagation through Structure
- Vision Example

Next Lecture:

- Matrix-Vector RNNs: Relation classification
- Recursive Neural Tensor Networks: Sentiment Analysis
- Tree LSTMs: Phrase Similarity

Building on Word Vector Space Models



the country of my birth
the place where I was born

But how can we represent the meaning of longer phrases?

By mapping them into the same vector space!

Semantic Vector Spaces

Vectors representing
Phrases and Sentences
that do not ignore word order
and capture semantics for NLP tasks

Single Word Vectors

- Distributional Techniques
- Brown Clusters
- Useful as features inside models, e.g. CRFs for NER, etc.
- Cannot capture longer phrases

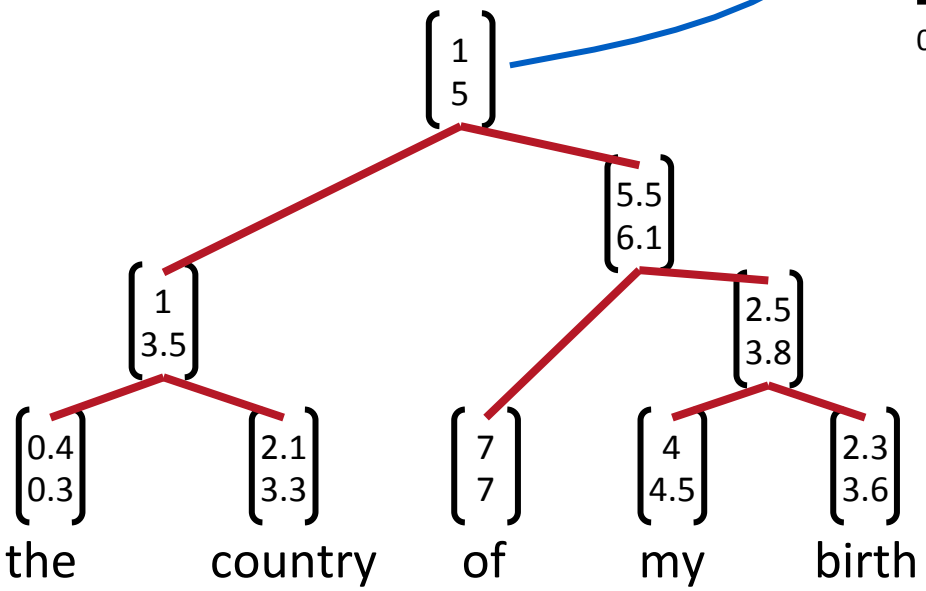
Documents Vectors

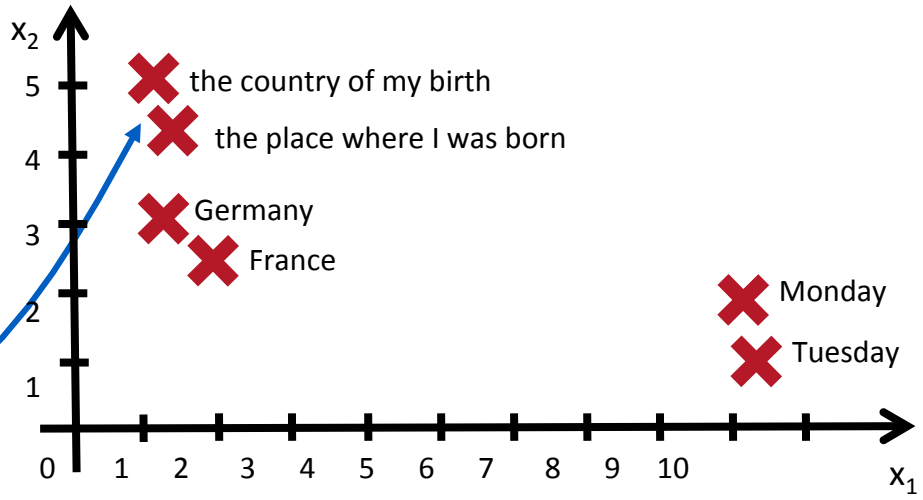
- Bag of words models
- PCA (LSA, LDA)
- Great for IR, document exploration, etc.
- Ignore word order, no detailed understanding

How should we map phrases into a vector space?

Use principle of compositionality

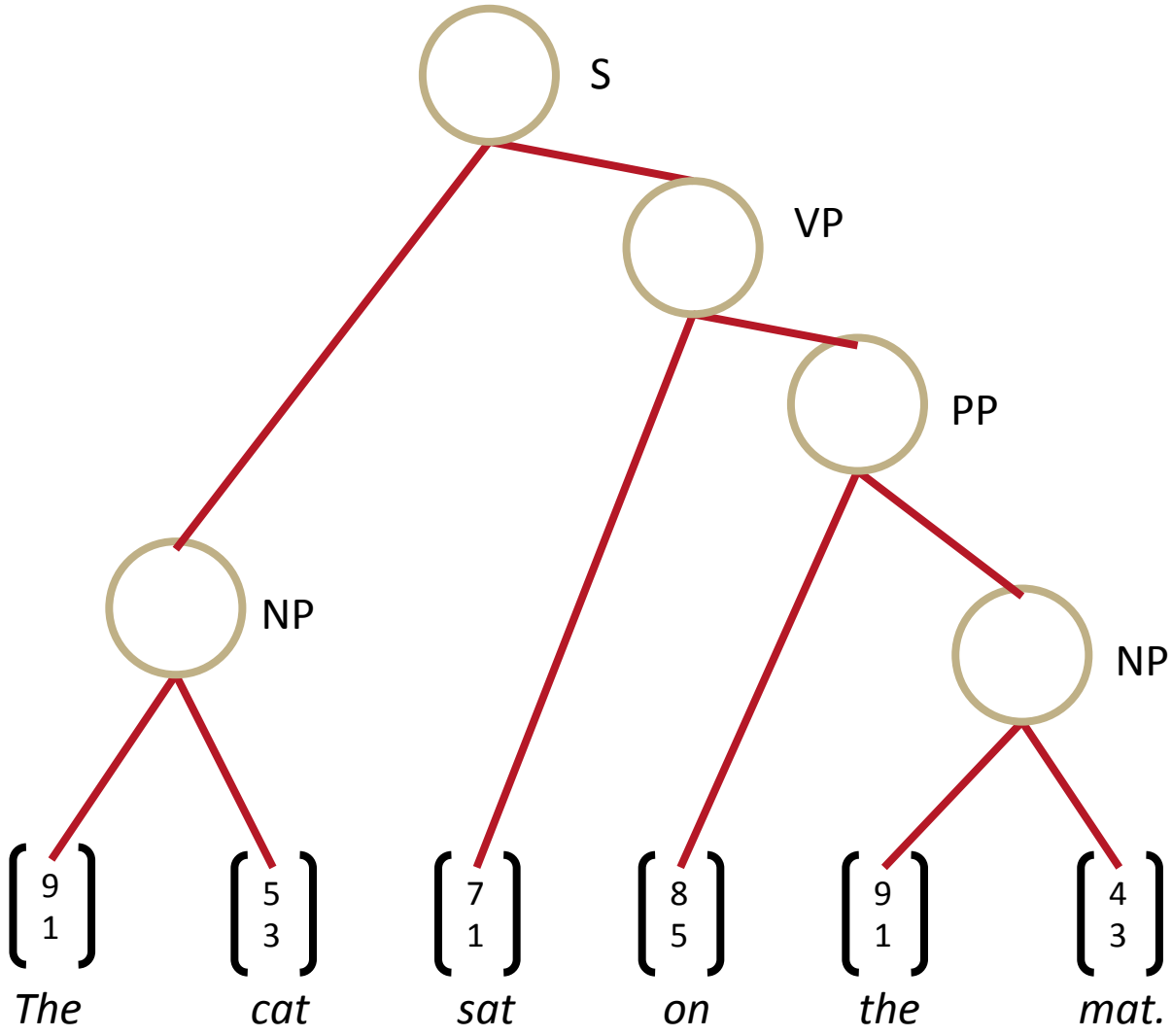
The meaning (vector) of a sentence is determined by
(1) the meanings of its words and
(2) the rules that combine them.



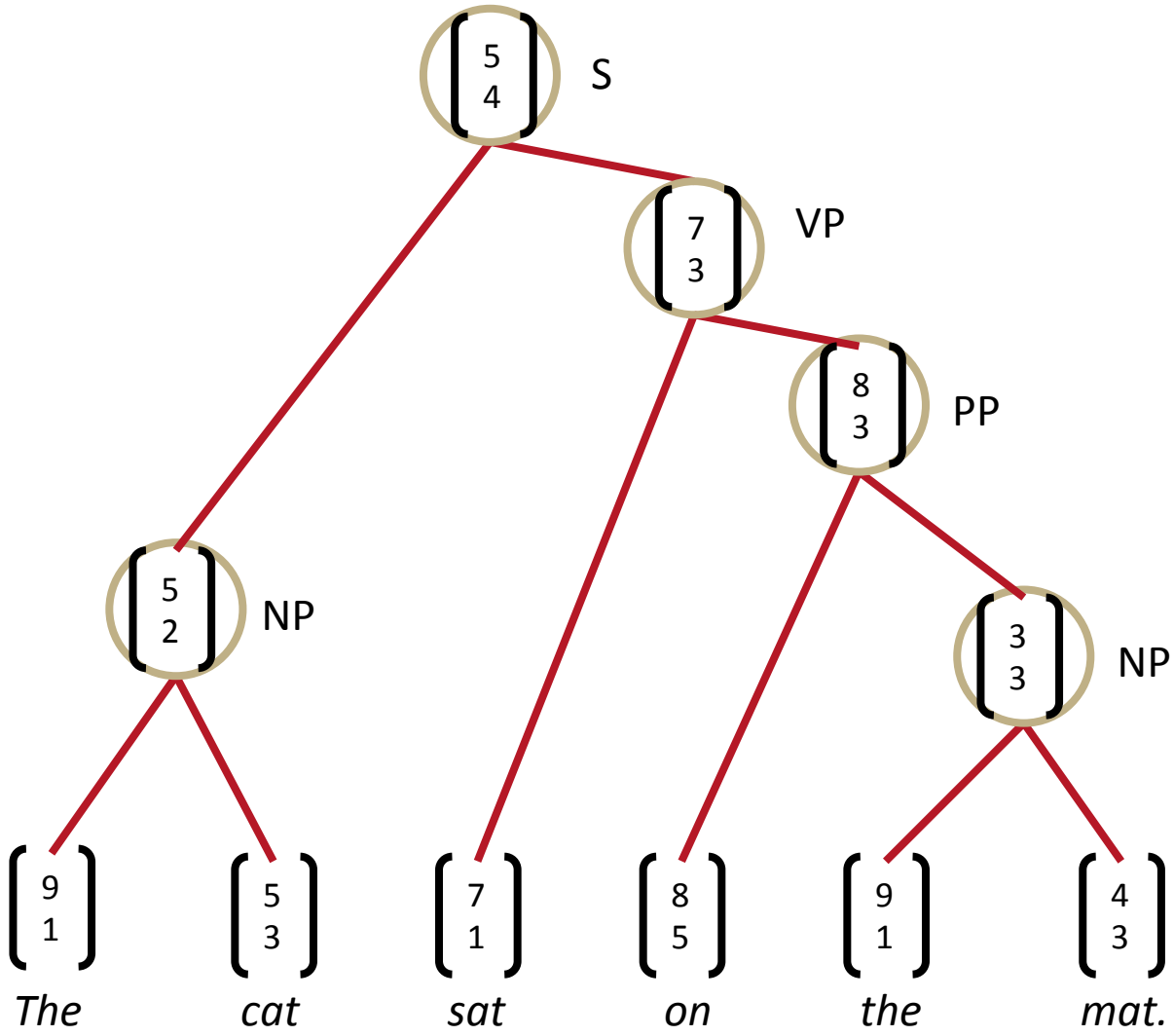


Models in this section can jointly learn parse trees and compositional vector representations

Sentence Parsing: What we want



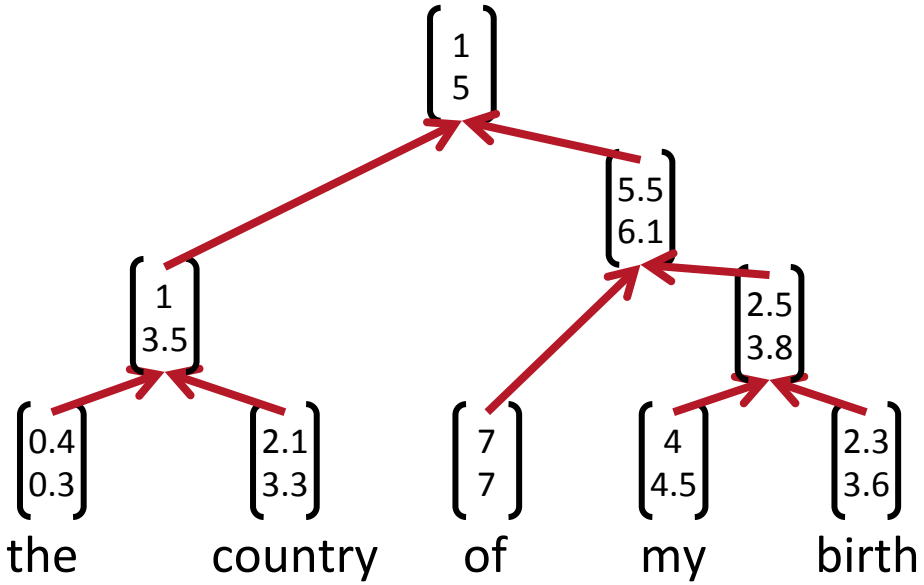
Learn Structure and Representation

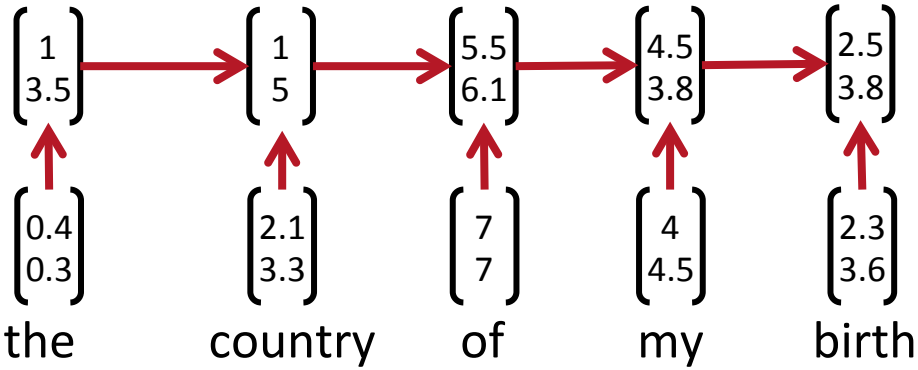


Learn Structure and Representation?

- Do we really need to learn this structure?

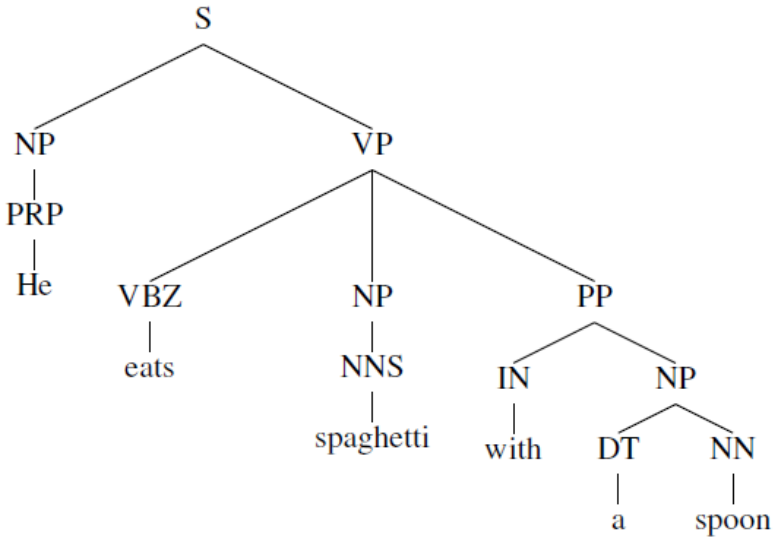
Sidenote: Recursive vs recurrent neural networks

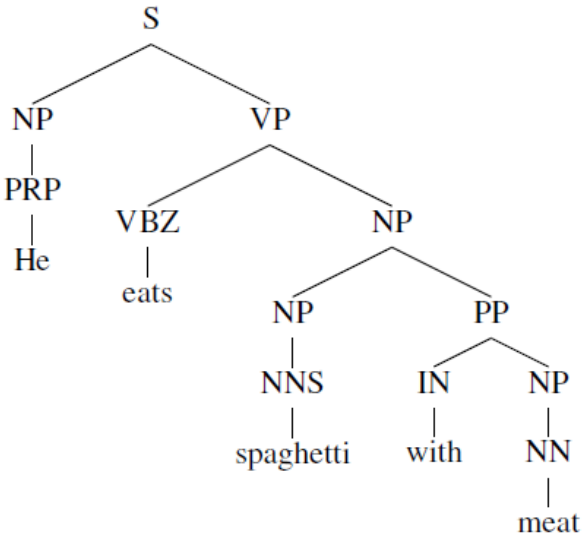




Sidenote: Are languages recursive?

- Cognitively debatable
- But: recursion helpful in describing natural language
- Example: “the church which has nice windows”, a noun phrase containing a relative clause that contains a noun phrases
- Arguments for now: 1) Helpful in disambiguation:





Sidenote: Are languages recursive?

2) Helpful for some tasks to refer to specific phrases:

- John and Jane went to a big festival. They enjoyed the trip and the music there.
- "they": John and Jane
- "the trip": went to a big festival
- "there": big festival

3) Labeling less clear if specific to only subphrases

- I liked the bright screen but not the buggy slow keyboard of the phone. It was a pain to type with. It was nice to look at.

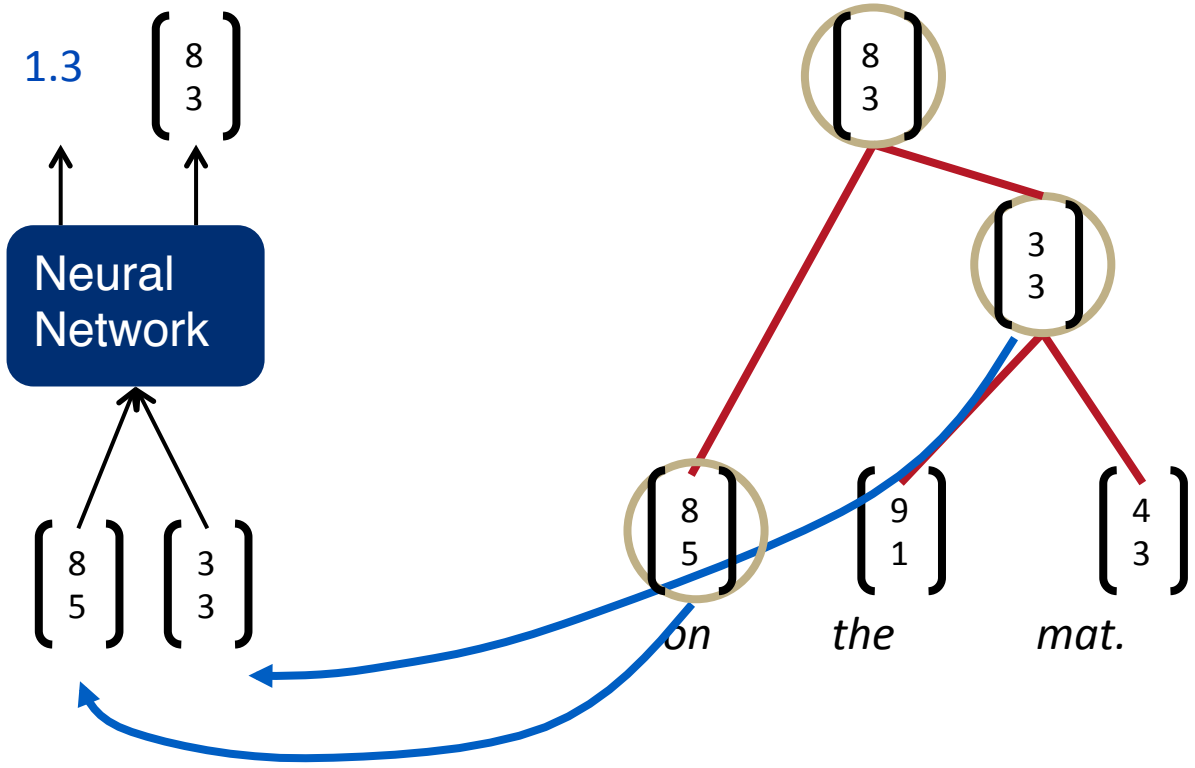
4) Works better for some tasks to use grammatical tree structure

- This is still up for debate.

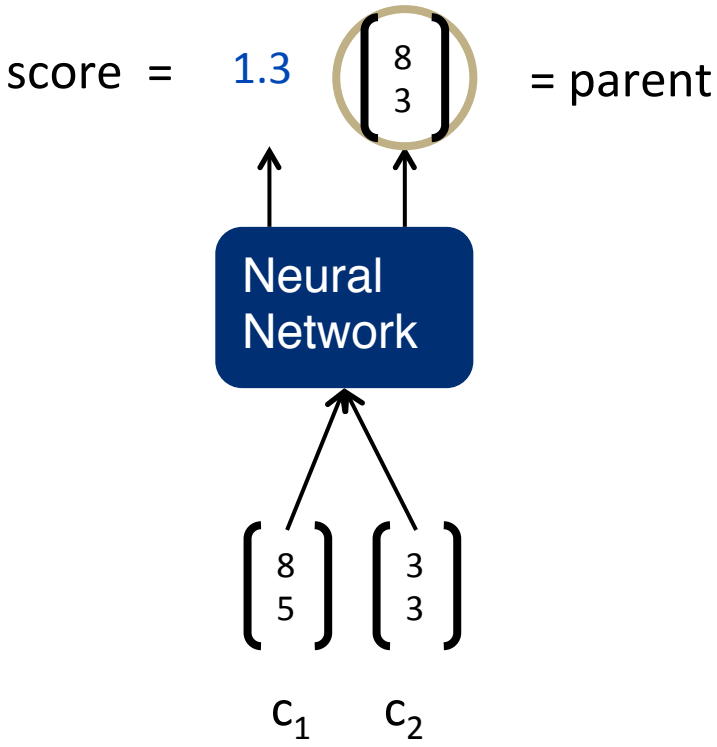
Recursive Neural Networks for Structure Prediction

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.



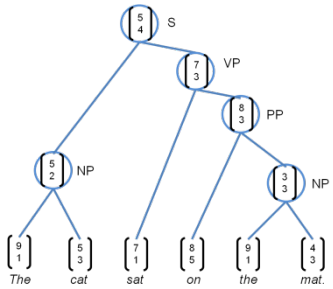
Recursive Neural Network Definition



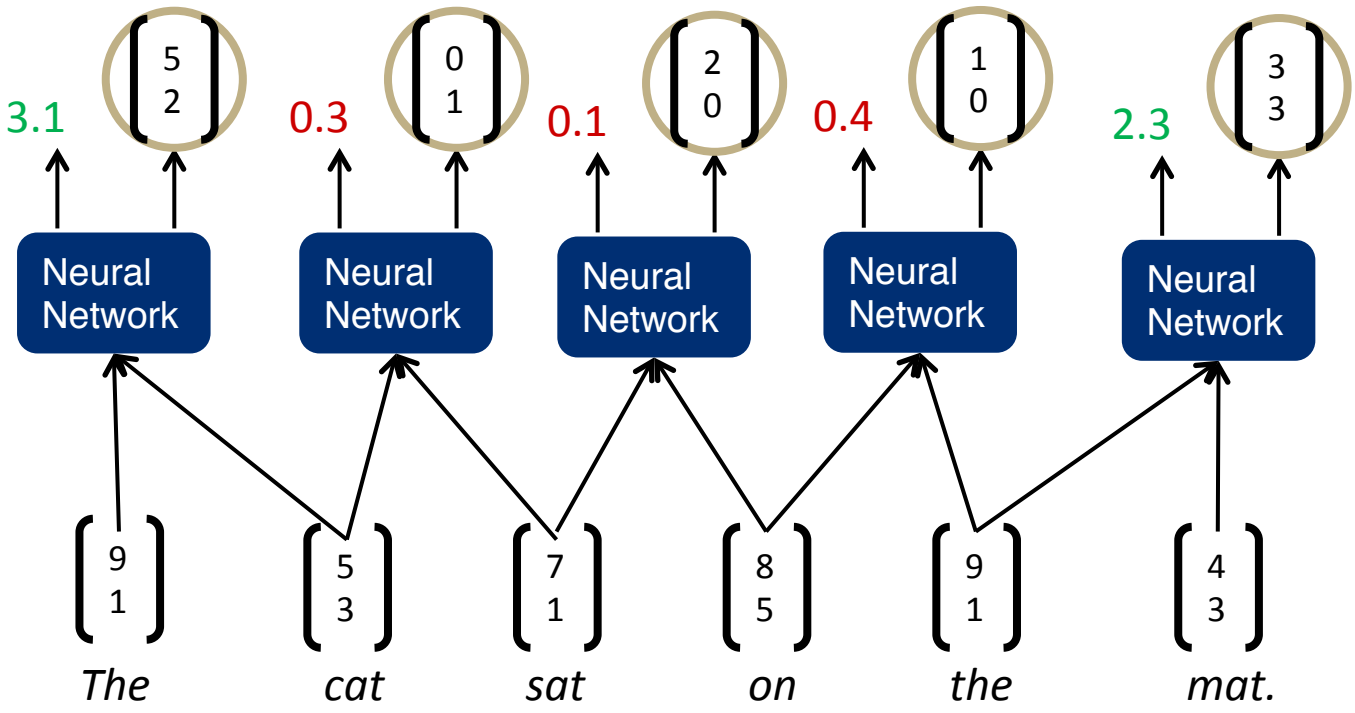
score = $U^T p$

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

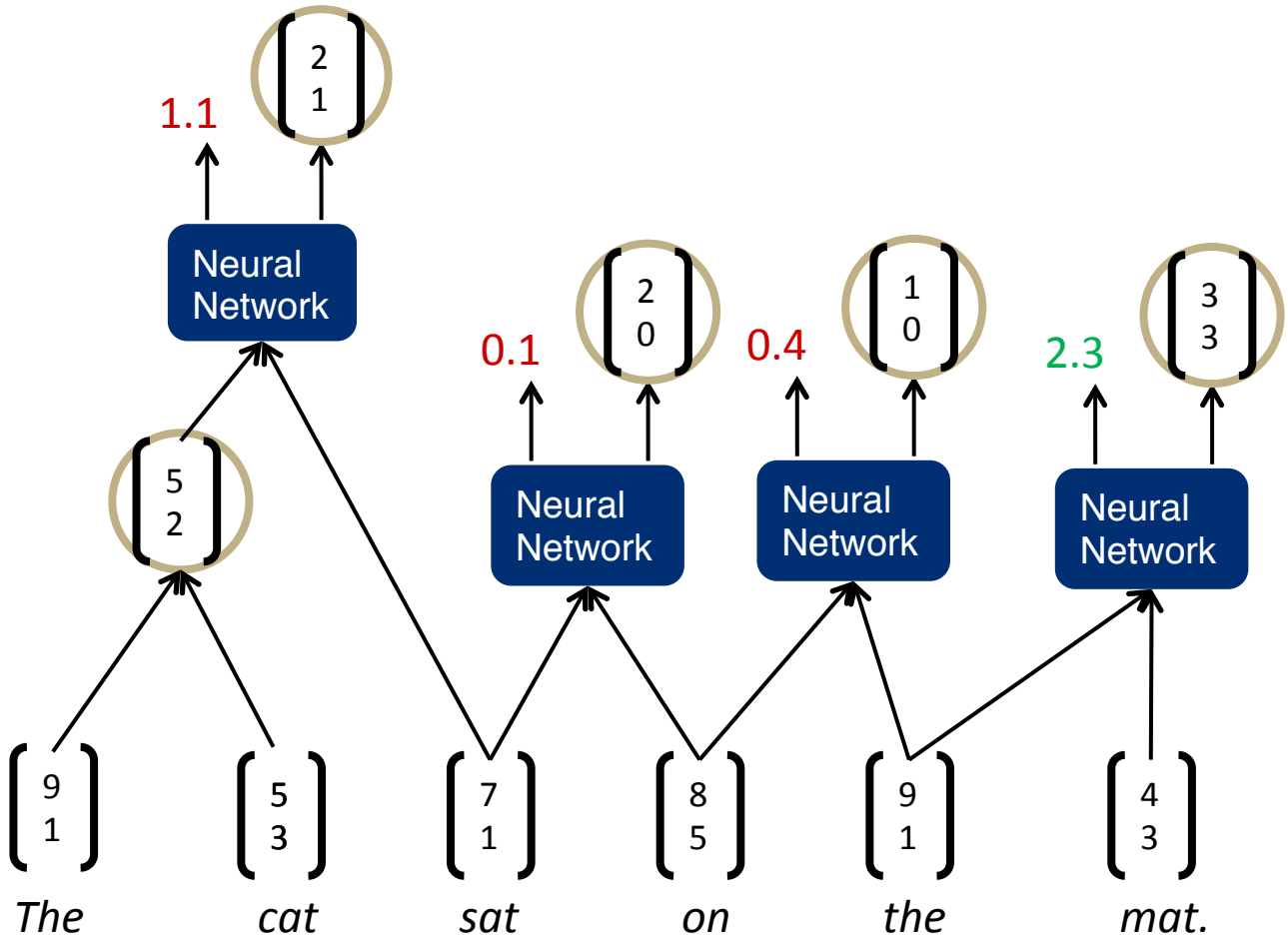
Same W parameters at all nodes of the tree



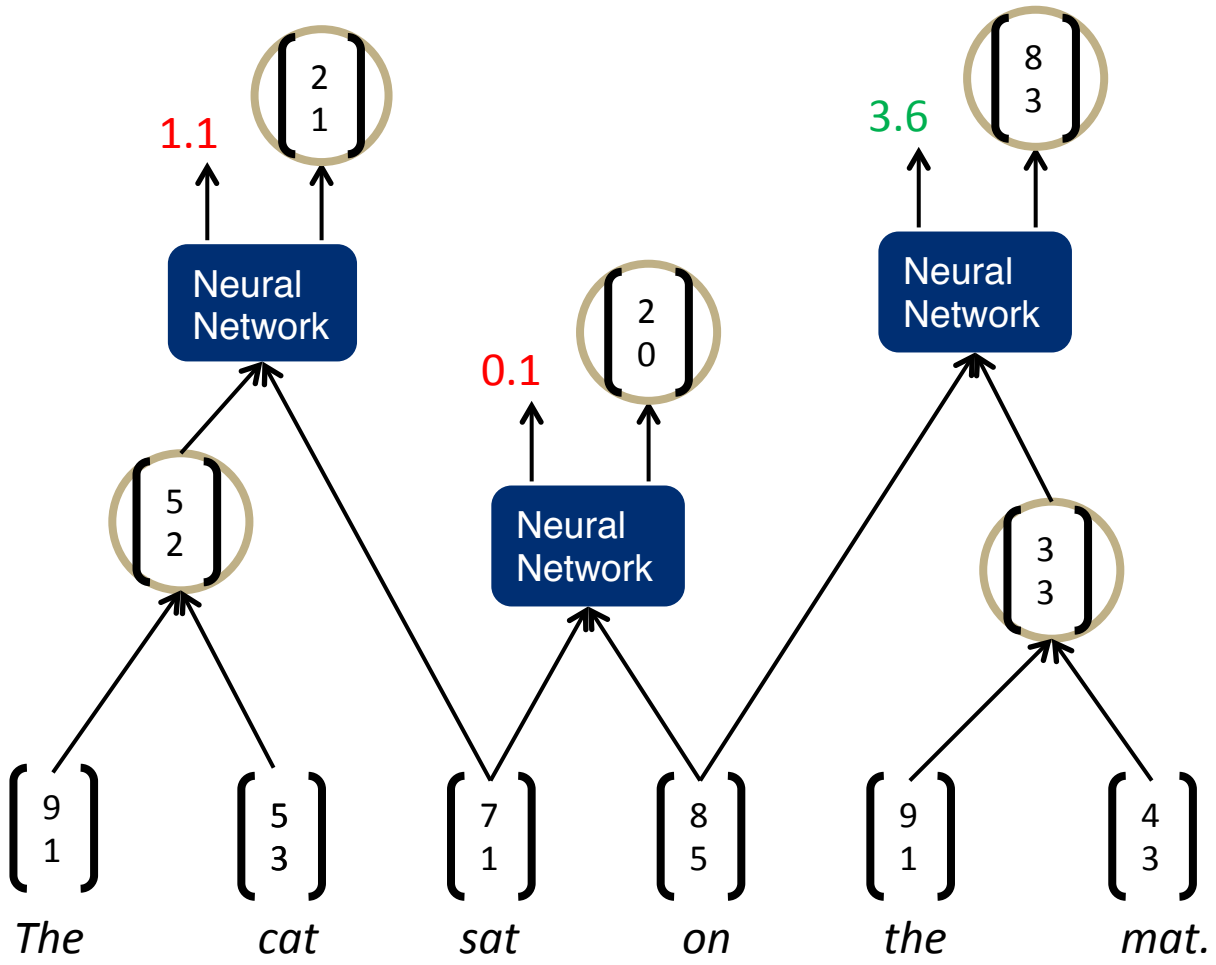
Parsing a sentence with an RNN



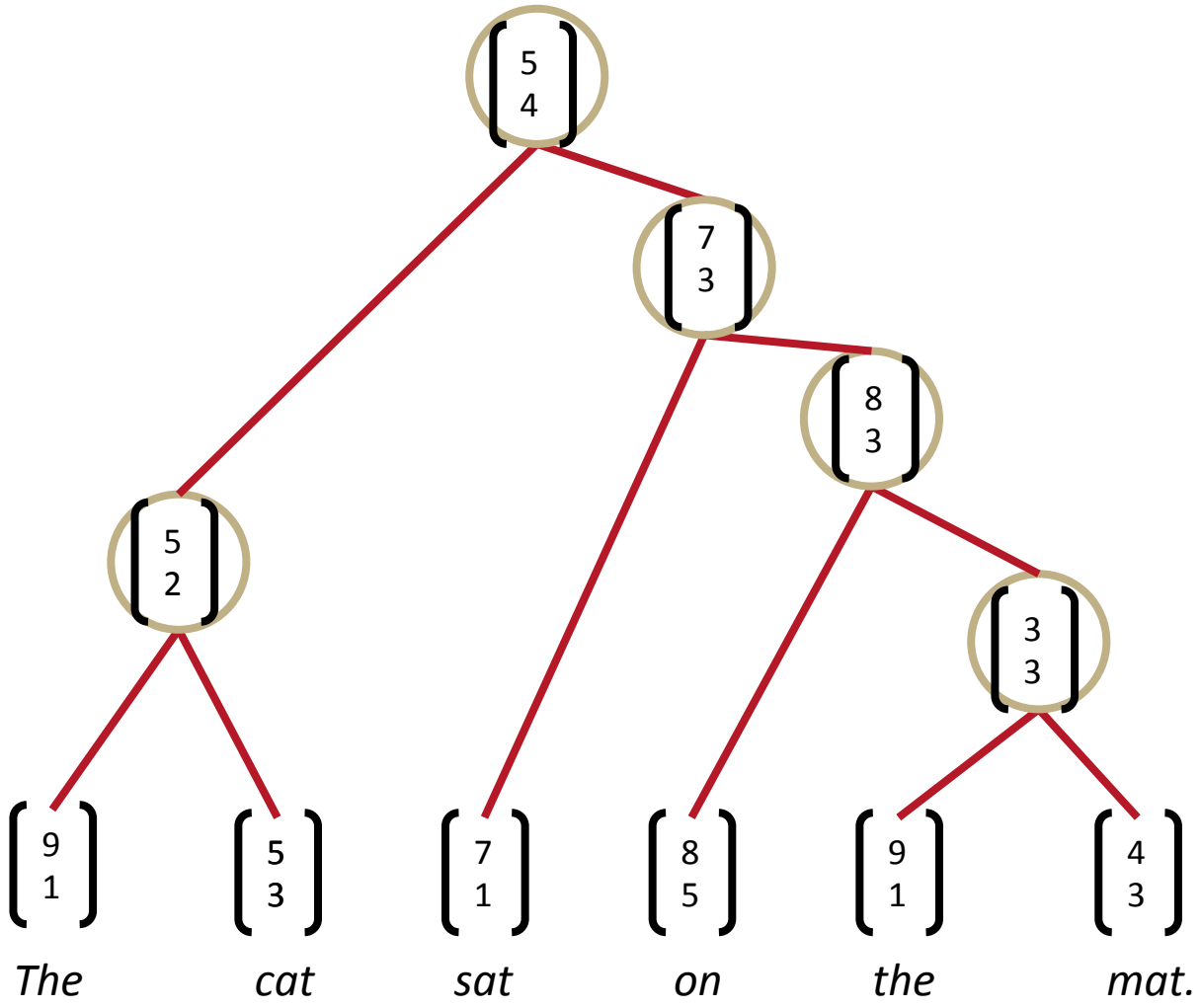
Parsing a sentence



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:

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

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 maximally greedy
  - Instead: Beam Search with Chart

Backpropagation Through Structure

Introduced by Goller & Küchler (1996)

Principally the same as general backpropagation

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

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

Three differences resulting from the recursion and tree structure:

1. Sum derivatives of *W* from all nodes
2. Split derivatives at each node
3. Add error messages

BTS: 1) Sum derivatives of all nodes

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

Intuition via example:

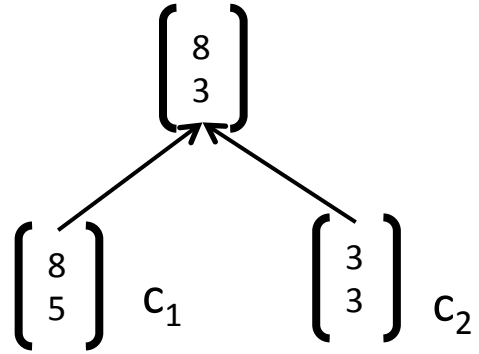
$$\begin{aligned}
& \frac{\partial}{\partial W} f(W(f(Wx)) \\
= \;& f'(W(f(Wx)) \left(\left(\frac{\partial}{\partial W} W \right) f(Wx) + W \frac{\partial}{\partial W} f(Wx) \right) \\
= \;& f'(W(f(Wx)) \left(f(Wx) + W f'(Wx) x \right)
\end{aligned}$$

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

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

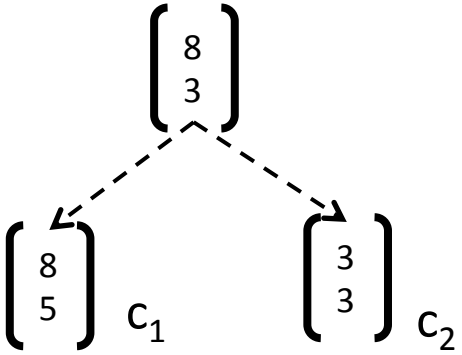
BTS: 2) Split derivatives at each node

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:

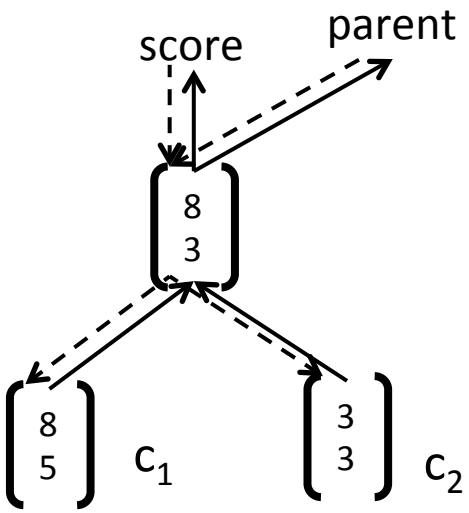


where each child's error is *n*-dimensional

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

BTS: 3) Add error messages

- At each node:
  - What came up (fprop) must come down (bprop)
  - Total error messages δ = 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:])
```

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

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

BTS: Optimization

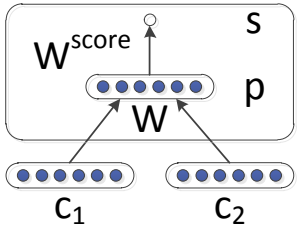
- As before, we can plug the gradients into a standard off-the-shelf L-BFGS optimizer or SGD
- Best results with AdaGrad (Duchi et al, 2011):

$$\theta_{t,i} = \theta_{t-1,i} - \frac{\alpha}{\sqrt{\sum_{\tau=1}^{t} g_{\tau,i}^2}} g_{t,i}$$

- For non-continuous objective use subgradient *method* (Ratliff et al. 2007)

Discussion: Simple RNN

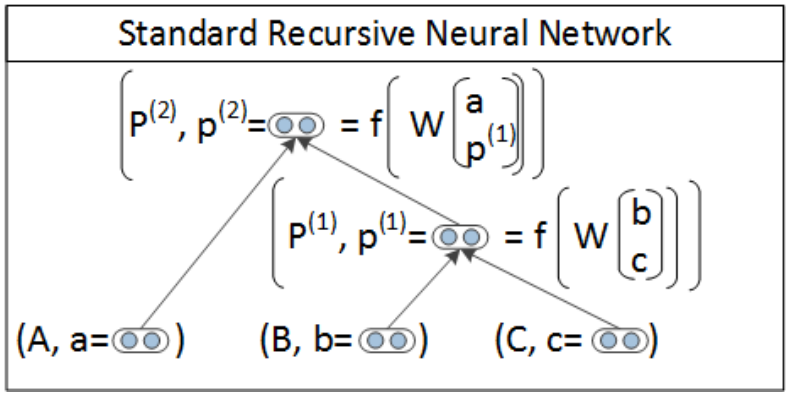
- Good results with single matrix RNN (more later)
- Single weight matrix RNN could capture some phenomena but not adequate for more complex, higher order composition and parsing long sentences

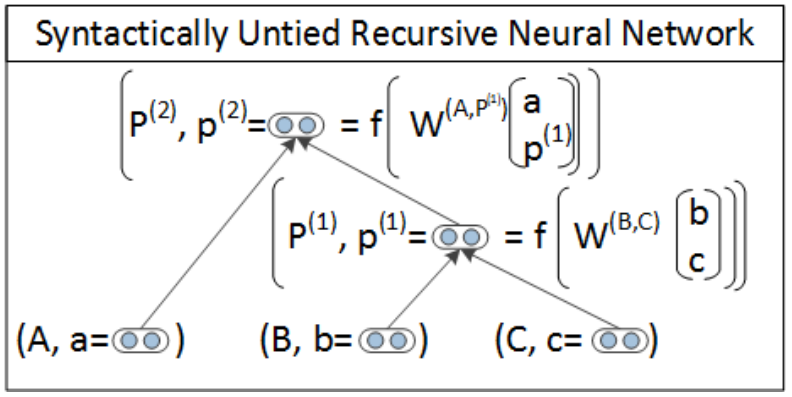


- The composition function is the same for all syntactic categories, punctuation, etc

Solution: Syntactically-Untied RNN

- Idea: Condition the composition function on the syntactic categories, "untie the weights"
- Allows for different composition functions for pairs of syntactic categories, e.g. Adv + AdjP, VP + NP
- Combines discrete syntactic categories with continuous semantic information





Solution: 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 Grammars: CVG = PCFG + RNN

Details: Compositional Vector Grammar

- Scores at each node computed by combination of PCFG and SU-RNN:

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

- Interpretation: Factoring discrete and continuous parsing in one model:

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

- Socher et al. (2013)

Related work for recursive neural networks

Pollack (1990): Recursive auto-associative memories

Previous Recursive Neural Networks work by Goller & Küchler (1996), Costa et al. (2003) assumed fixed tree structure and used one hot vectors.

Hinton (1990) and Bottou (2011): Related ideas about recursive models and recursive operators as smooth versions of logic operations

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

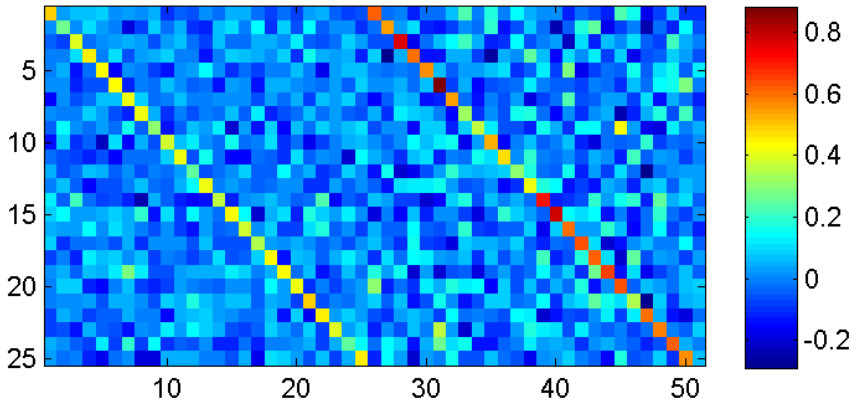
Experiments

- 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, 20% faster than Stanford factored parser

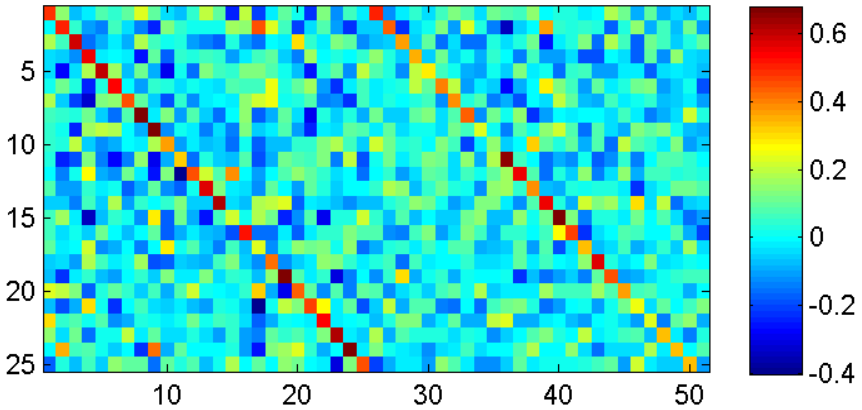
Parser	Test, All Sentences
Stanford PCFG, (Klein and Manning, 2003a)	85.5
Stanford Factored (Klein and Manning, 2003b)	86.6
Factored PCFGs (Hall and Klein, 2012)	89.4
Collins (Collins, 1997)	87.7
SSN (Henderson, 2004)	89.4
Berkeley Parser (Petrov and Klein, 2007)	90.1
CVG (RNN) (Socher et al., ACL 2013)	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 Analysis

- Learns notion of soft head words



DT-NP



VP-NP

Analysis of resulting vector representations

All the figures are adjusted for seasonal variations

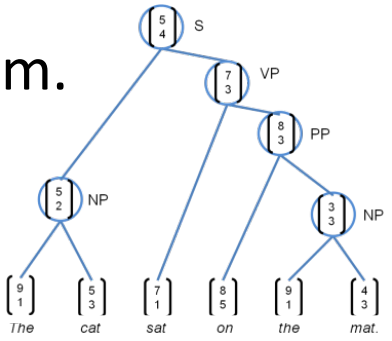
1. All the numbers are adjusted for seasonal fluctuations
2. All the figures are adjusted to remove usual seasonal patterns

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.

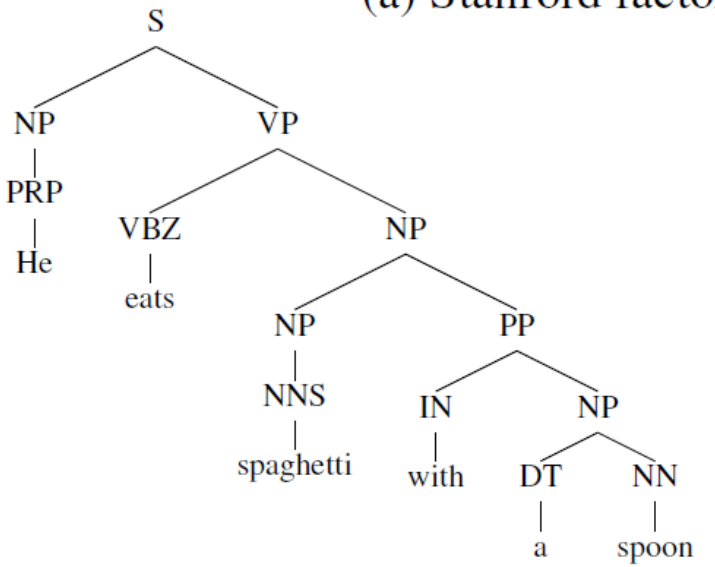


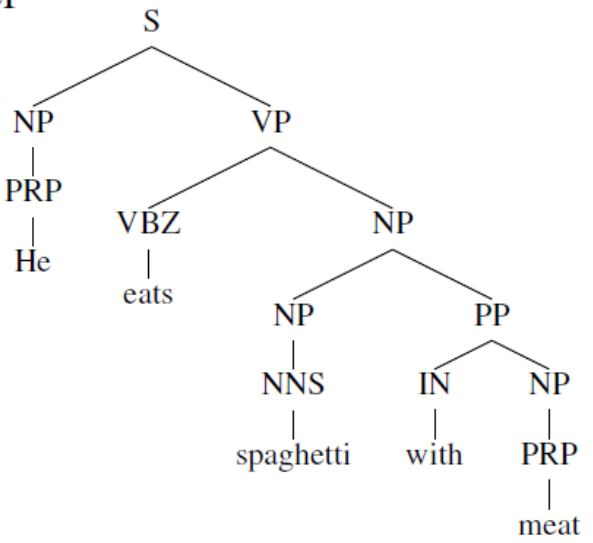
SU-RNN Analysis

- Can transfer semantic information from single related example
- Train sentences:
  - He eats spaghetti with a fork.
  - She eats spaghetti with pork.
- Test sentences
  - He eats spaghetti with a spoon.
  - He eats spaghetti with meat.

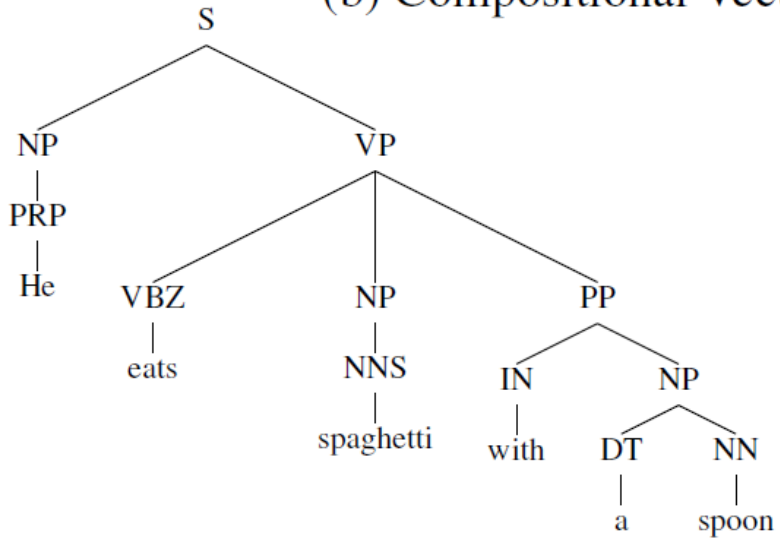
SU-RNN Analysis

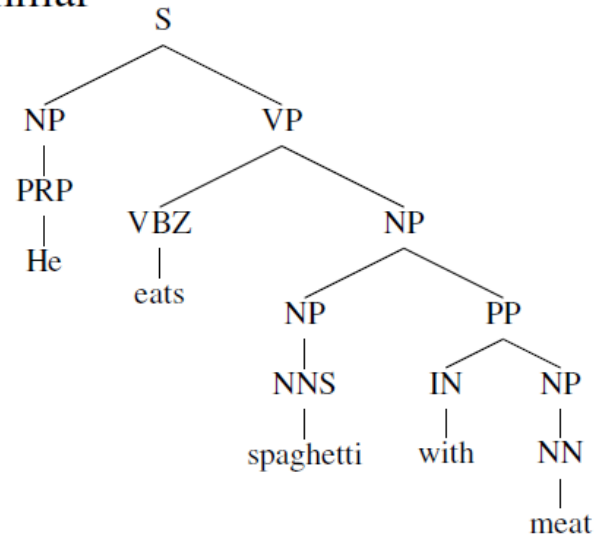
(a) Stanford factored parser





(b) Compositional Vector Grammar



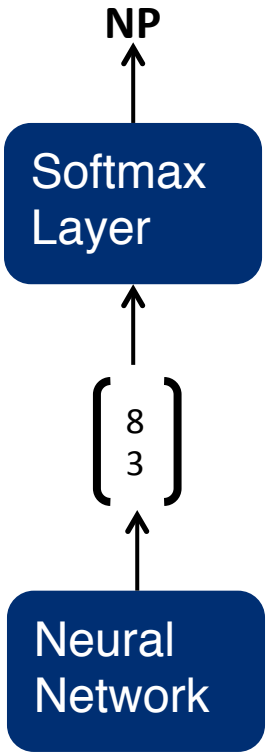


Labeling in Recursive Neural Networks

- We can use each node's representation as features for a *softmax* classifier:

$$p(c|p) = softmax(Sp)$$

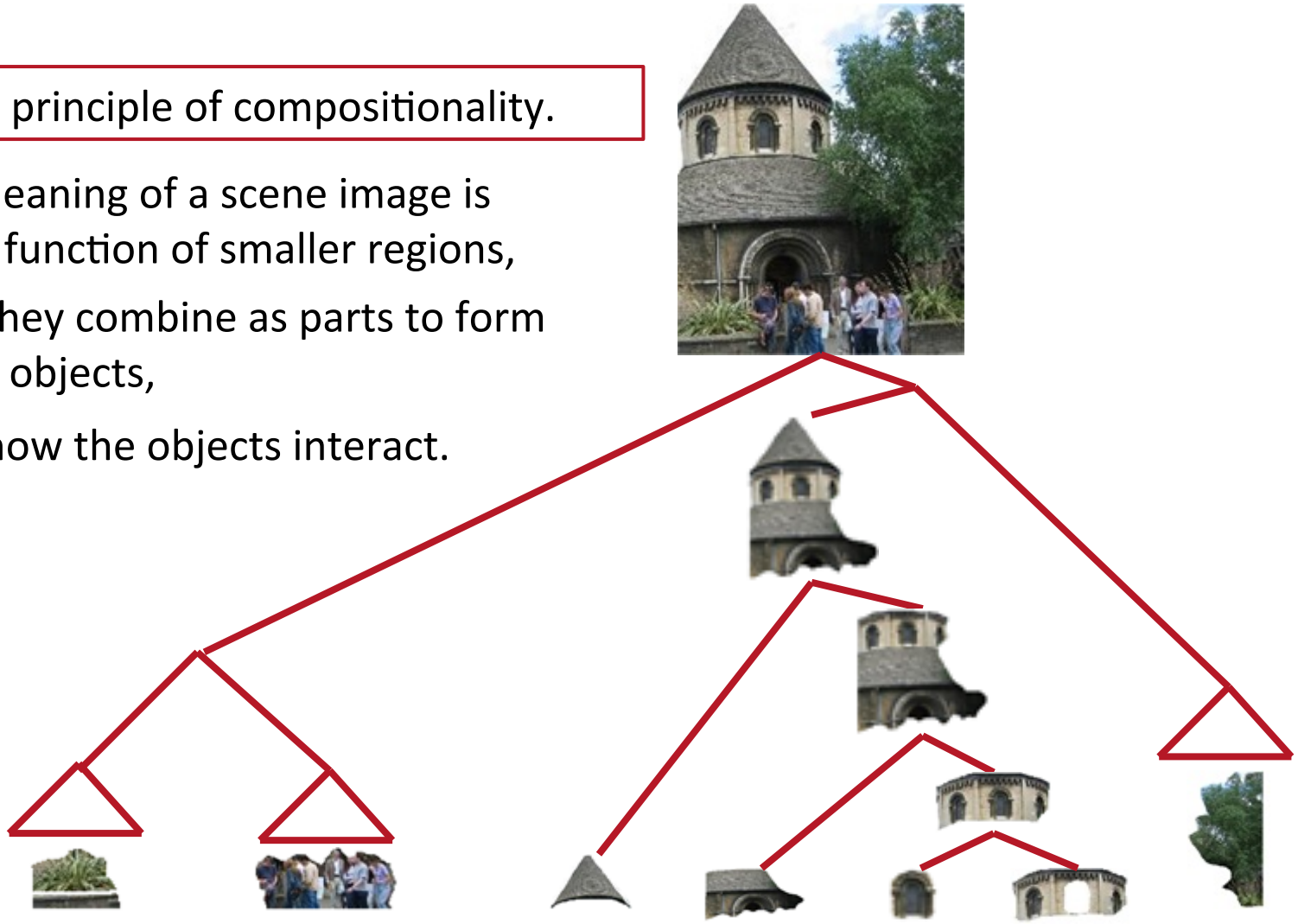
- Training similar to model in part 1 with standard cross-entropy error + scores



Scene Parsing

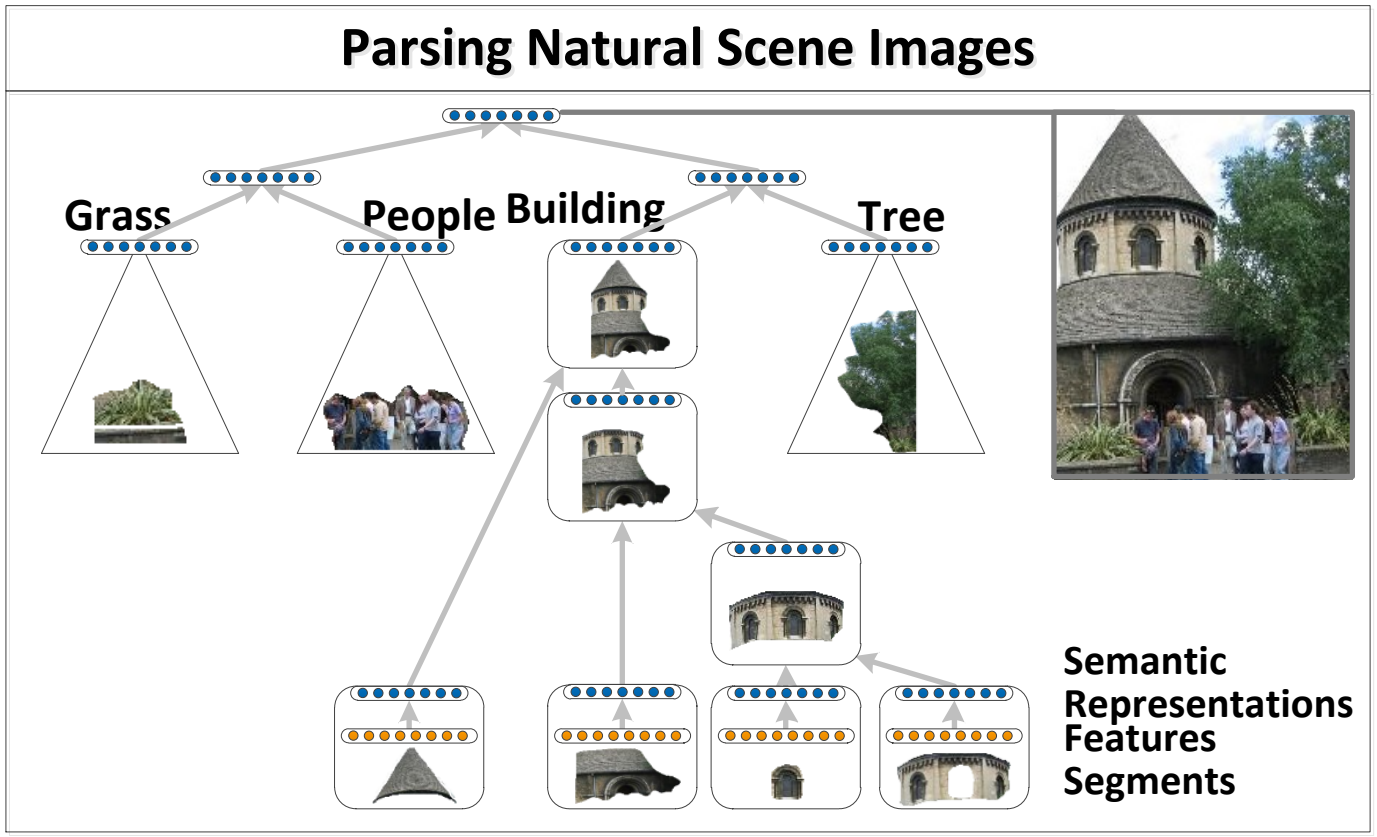
Similar principle of compositionality.

- 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.

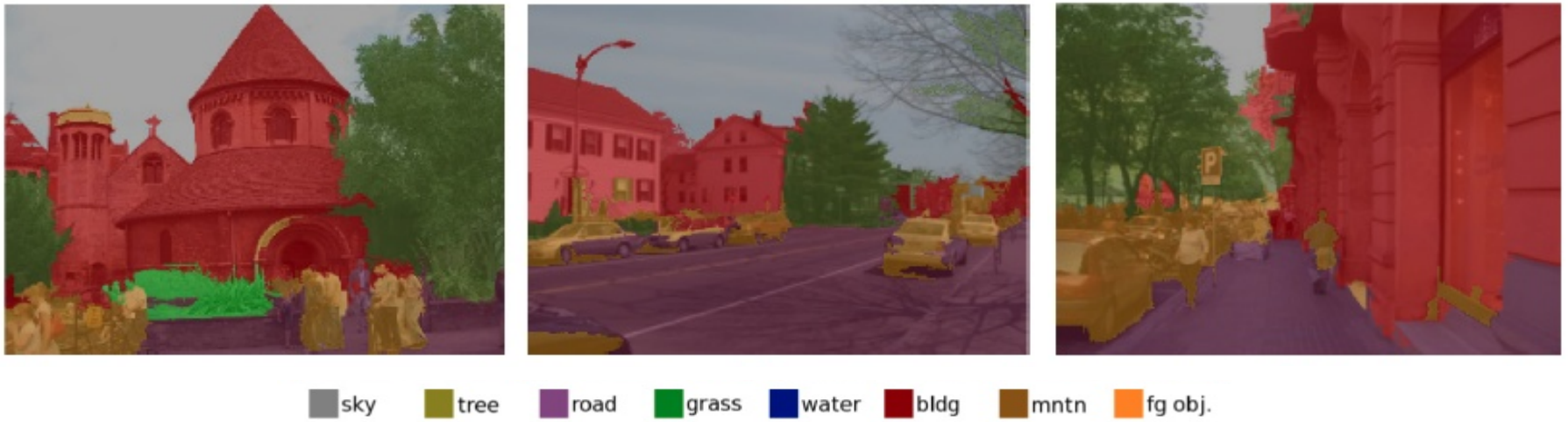


Algorithm for Parsing Images

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)