Lecture 14: Tree Recursive Neural Networks and Constituency Parsing
Lecture Plan

1. Motivation: Compositionality and Recursion
2. Structure prediction with simple Tree RNN: Parsing
4. Backpropagation through Structure
5. More complex units

Reminders/comments:

Learn up on GPUs, Azure, Docker
Ass 4: Get something working, using a GPU for milestone
Final project discussions – come meet with us!
1. The spectrum of language in CS
Semantic interpretation of language – Not just word vectors

How can we know when larger units are similar in meaning?

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

Marc D. Hauser,¹* Noam Chomsky,² W. Tecumseh Fitch¹

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
- But: recursion is natural for describing language
- *The man from [the company that you spoke with about [the project] yesterday]]*
- noun phrase containing a noun phrase containing a noun phrase
- Arguments for now: 1) Helpful in disambiguation:
Is recursion useful?

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) Works better for some tasks to use grammatical tree structure
   - It’s a powerful prior for language structure
Building on Word Vector Space Models

By mapping them into the same vector space!

How can we represent the meaning of longer phrases?

the country of my birth
the place where I was born

By mapping them into the same vector space!
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
Constituency Sentence Parsing: What we want

The cat sat on the mat.
The cat sat on the mat.
Recursive vs. recurrent neural networks
Recursive vs. recurrent neural networks

- Recursive neural nets require a parser to get tree structure

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

- RNN: Get compositional vectors for grammatical phrases only
- CNN: Computes vectors for every possible phrase
- Example: “the country of my birth” computes vectors for:
  - the country, country of, of my, my birth, the country of, country of my, of my birth, the country of my, country of my birth
- Regardless of whether each is grammatical – many don’t make sense
- Don’t need parser
- But maybe not very linguistically or cognitively plausible
Relationship between RNNs and CNNs

- CNN

- RNN
Relationship between RNNs and CNNs

- CNN

- RNN

people there speak slowly

people there speak slowly

Richard Socher 3/2/17
2. 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 = $U^T p$

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

Same $W$ parameters at all nodes of the tree

score = 1.3

= parent
The cat sat on the mat.
The cat sat on the mat.
The cat sat on the mat.
The cat sat on the mat.
Max-Margin Framework - Details

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

\[ s(x, y) = \sum_{n \in \text{nodes}(y)} s_n \]

- \( x \) is sentence; \( y \) is parse tree
Max-Margin Framework - Details

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

\[ J = \sum_i s(x_i, y_i) - \max_{y \in A(x_i)} (s(x_i, y) + \Delta(y, y_i)) \]

• 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
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 (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:

\[
\frac{\partial}{\partial W} f(W(f(Wx))) = f'(W(f(Wx))) \left( \left( \frac{\partial}{\partial W} W \right) f(Wx) + W \frac{\partial}{\partial W} f(Wx) \right)
\]

\[
= f'(W(f(Wx))) (f(Wx) + W f'(Wx)x)
\]

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

\[
\frac{\partial}{\partial W_2} f(W_2(f(W_1x))) + \frac{\partial}{\partial W_1} f(W_2(f(W_1x)))
\]

\[
= f'(W_2(f(W_1x))) (f(W_1x)) + f'(W_2(f(W_1x))) (W_2 f'(W_1x)x)
\]

\[
= f'(W_2(f(W_1x))) (f(W_1x) + W_2 f'(W_1x)x)
\]

\[
= f'(W(f(Wx))) (f(Wx) + W f'(Wx)x)
\]
BTS: 2) Split derivatives at each node

During forward prop, the parent is computed using 2 children

\[
p = \tanh(W\begin{pmatrix} c_1 \\ c_2 \end{pmatrix} + b)
\]

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

where each child’s error is \( n \)-dimensional

\[
\delta_{p \rightarrow c_1 c_2} = [\delta_{p \rightarrow c_1} \delta_{p \rightarrow c_2}]
\]
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
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

```python
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)
Deep Reinforcement Learning for Dialogue Generation

Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky
Seq2Seq for Dialogue

Encode previous message(s) into vector

Decode vector into response

How are you I am fine
Seq2Seq for Dialogue

Encode previous message(s) into vector

Decode vector into response

Train by maximizing

\[ p(\text{response}|\text{input}) \]

where the response is produced by a human
Problems with Seq2Seq

How old are you?
Problems with Seq2Seq

How old are you?

16?

I’m 16
Problems with Seq2Seq

How old are you?

16?

I’m 16

You don’t know what you’re saying

I don’t know what you’re talking about

You don’t know what you’re saying

I don’t know what you’re talking about
Problems with Seq2Seq

- **How old are you?**
  - **16?**
- You don’t know what you’re saying
  - I don’t know what you’re talking about
- I’m 16
  - reasonable, but unhelpful
- I don’t know what you’re talking about
  - generic
probable response != good response
What is a good response?

- Reasonable
- Nonrepetitive
- Easy to answer
What is a good response?

- **Reasonable**: $p(\text{response}|\text{input})$ is high according to seq2seq model
- **Nonrepetitive**: similarity between response and previous messages is low
- **Easy to answer**: $p(\text{“i don’t know”}|\text{response})$ is low

Scoring function: $R(\text{response}) = \text{reasonable_score} + \text{nonrepetitive_score} + \text{easy\_to\_answer\_score}$
Reinforcement Learning

Learn from rewards instead of from examples

1. Encode input into a vector

How are you
Learn from rewards instead of from examples

2. Have the system generate a response
Reinforcement Learning

Learn from rewards instead of from examples

3. Receive reward $R(\text{response})$
   - Train system to maximize reward

$R = -5$
### Quantitative Results

<table>
<thead>
<tr>
<th>Setting</th>
<th>Gain</th>
</tr>
</thead>
<tbody>
<tr>
<td>single-turn general quality</td>
<td>0.02</td>
</tr>
<tr>
<td>single-turn ease to answer</td>
<td>0.12</td>
</tr>
<tr>
<td>multi-turn general quality</td>
<td>0.17</td>
</tr>
</tbody>
</table>
Qualitative Results

How old are you?

I thought you were 12

I’m 16. Why are you asking?

What made you think so?
Qualitative Results

How old are you?

I thought you were 12

I’m 16. Why are you asking?

What made you think so?

You don’t know what you’re saying

I don’t know what you’re talking about

You don’t know what you’re saying
Conclusion

• Reinforcement learning useful when we want our model to do more than produce a probable human label

• Many more application of RL to NLP!
  Information extraction, question answering, task-oriented dialogue, coreference resolution, and more
Discussion: Simple RNN

• 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.
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.
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
Details: Compositional Vector Grammar

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

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

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

\[ P((P_1, p_1) \rightarrow (B, b)(C, c)) = P(p_1 \rightarrow b \ c|P_1 \rightarrow B \ C)P(P_1 \rightarrow B \ C) \]

• 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

<table>
<thead>
<tr>
<th>Parser</th>
<th>Test, All Sentences</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stanford PCFG, (Klein and Manning, 2003a)</td>
<td>85.5</td>
</tr>
<tr>
<td>Stanford Factored (Klein and Manning, 2003b)</td>
<td>86.6</td>
</tr>
<tr>
<td>Factored PCFGs (Hall and Klein, 2012)</td>
<td>89.4</td>
</tr>
<tr>
<td>Collins (Collins, 1997)</td>
<td>87.7</td>
</tr>
<tr>
<td>SSN (Henderson, 2004)</td>
<td>89.4</td>
</tr>
<tr>
<td>Berkeley Parser (Petrov and Klein, 2007)</td>
<td>90.1</td>
</tr>
<tr>
<td>CVG (RNN) (Socher et al., ACL 2013)</td>
<td>85.0</td>
</tr>
<tr>
<td>CVG (SU-RNN) (Socher et al., ACL 2013)</td>
<td>90.4</td>
</tr>
<tr>
<td>Charniak - Self Trained (McClosky et al. 2006)</td>
<td>91.0</td>
</tr>
<tr>
<td>Charniak - Self Trained-ReRanked (McClosky et al. 2006)</td>
<td>92.1</td>
</tr>
</tbody>
</table>
SU-RNN / CVG [Socher, Bauer, Manning, Ng 2013]

Learns soft notion of head words

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

\[ p(c|p) = \text{softmax}(Sp) \]

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

\[ very \text{ good} \]

Proposal: A new composition function
Version 3: Matrix-vector RNNs

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

\[ p = f \left( W \begin{bmatrix} a \\ b \end{bmatrix} \right) \]

\[ p = f \left( W \begin{bmatrix} Ba \\ Ab \end{bmatrix} \right) \]
Compositionality Through Recursive Matrix-Vector Recursive Neural Networks

\[ p = \tanh(W \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} + b) \]
Matrix-vector RNNs
[Socher, Huval, Bhat, Manning, & Ng, 2012]

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

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

\[ W_M \in \mathbb{R}^{n \times 2n} \]
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} \rightarrow \text{component-whole relationship (e2,e1)}

• Build a single compositional semantics for the minimal constituent including both terms
## Classification of Semantic Relationships

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Features</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM</td>
<td>POS, stemming, syntactic patterns</td>
<td>60.1</td>
</tr>
<tr>
<td>MaxEnt</td>
<td>POS, WordNet, morphological features, noun compound system, thesauri, Google n-grams</td>
<td>77.6</td>
</tr>
<tr>
<td>SVM</td>
<td>POS, WordNet, prefixes, morphological features, dependency parse features, Levin classes, PropBank, FrameNet, NomLex-Plus, Google n-grams, paraphrases, TextRunner</td>
<td>82.2</td>
</tr>
<tr>
<td>RNN</td>
<td>–</td>
<td>74.8</td>
</tr>
<tr>
<td>MV-RNN</td>
<td>–</td>
<td>79.1</td>
</tr>
<tr>
<td>MV-RNN</td>
<td>POS, WordNet, NER</td>
<td>82.4</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Method</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pixel CRF (Gould et al., ICCV 2009)</td>
<td>74.3</td>
</tr>
<tr>
<td>Classifier on superpixel features</td>
<td>75.9</td>
</tr>
<tr>
<td>Region-based energy (Gould et al., ICCV 2009)</td>
<td>76.4</td>
</tr>
<tr>
<td>Local labelling (Tighe &amp; Lazebnik, ECCV 2010)</td>
<td>76.9</td>
</tr>
<tr>
<td>Superpixel MRF (Tighe &amp; Lazebnik, ECCV 2010)</td>
<td>77.5</td>
</tr>
<tr>
<td>Simultaneous MRF (Tighe &amp; Lazebnik, ECCV 2010)</td>
<td>77.5</td>
</tr>
<tr>
<td>Recursive Neural Network</td>
<td>78.1</td>
</tr>
</tbody>
</table>

Stanford Background Dataset (Gould et al. 2009)
QCD-Aware Recursive Neural Networks for Jet Physics
Gilles Louppe, Kyunghun Cho, Cyril Becot, Kyle Cranmer