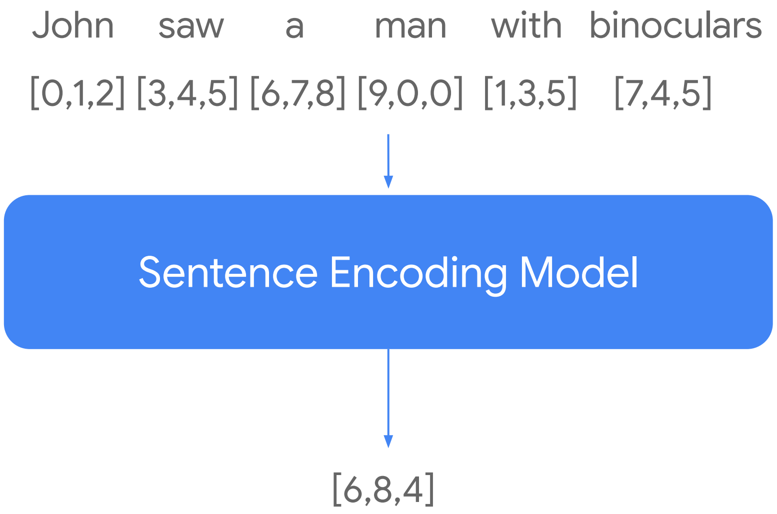
Modeling Hierarchical Structures with

Continuous Recursive Neural Networks

Jishnu Ray Chowdhury and Cornelia Caragea

Computer Science
University of Illinois at Chicago

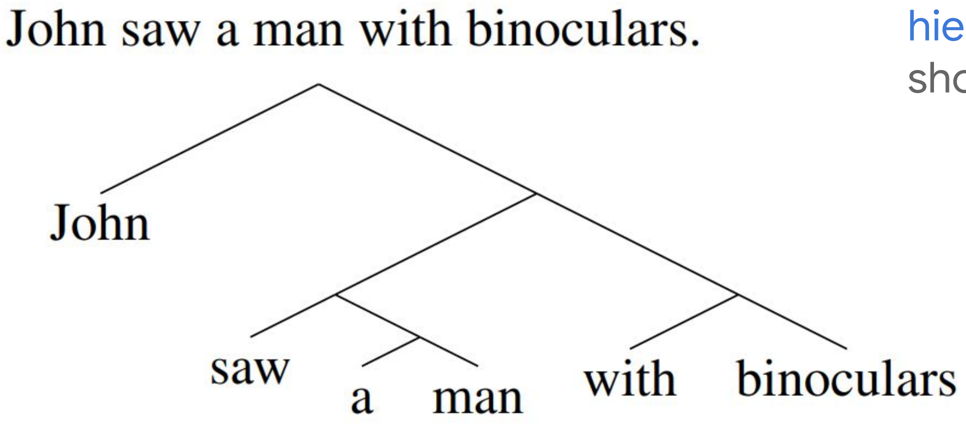
Sentence Encoding



- Many natural language processing tasks require the composition of a sequence of word vectors into a single sentence vector representing the *"meaning of the whole"*.

- Examples of such tasks:
    - Sentence Similarity,
    - Paraphrase mining,
    - Natural Language Inference,
    - Classification.

Hierarchies within Text

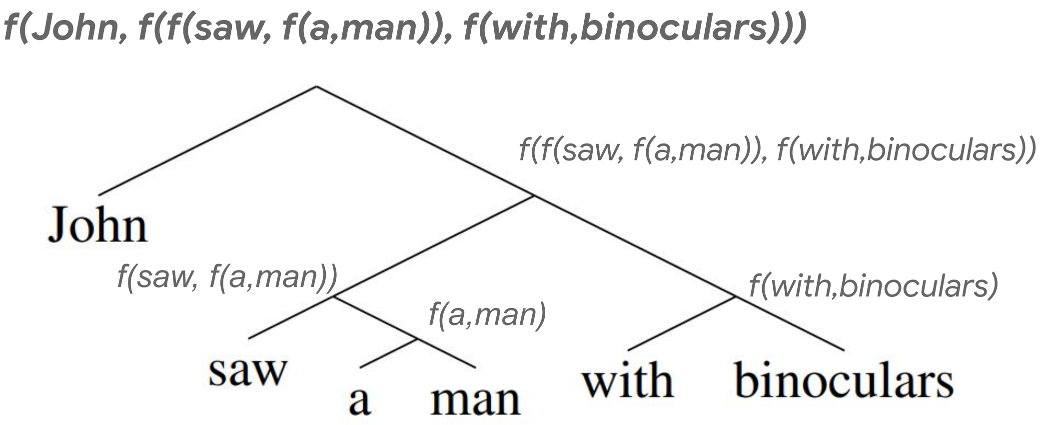


- Intuitively, understanding and modeling the hierarchical constituency structures in text should be useful for sentence composition.

Figure from: WooJin Chung and Samuel R Bowman. (2018). The lifted matrix-space model for semantic composition. In Proceedings of the 22nd Conference on Computational Natural Language Learning (CoNLL 2018),

Modeling Hierarchical Structures

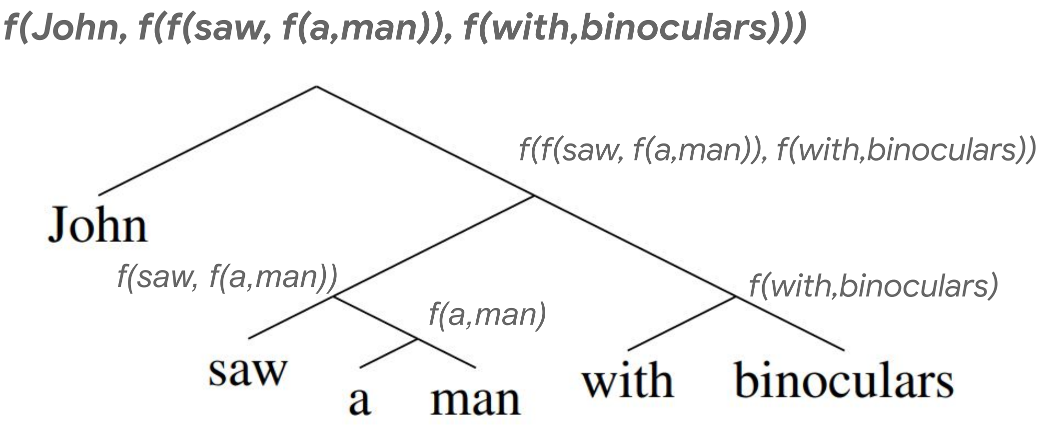
One way: Recursive Neural Networks (RvNNs)



- *f()* is a recursive composition function.

Modeling Hierarchical Structures

One way: **Recursive Neural Networks (RvNNs)**



- *f()* is a recursive composition function.

Limitation: Cannot learn the structure itself (structure needs to be provided as an input).

Latent Structure Learning Models

Reinforcement Learning or Biased Gradients

Chart Parsers

Stack Augmented Recurrent Neural Networks

Latent Structure Learning Models

Reinforcement Learning or Biased Gradients [1,2,3]

- Can increase variance or bias unless care is taken.
- Some of the models fail in simple synthetic tasks.[4]
- Sometimes use a single discrete structural merging decision per iteration.[2]

[1] “Learning to Compose Words into Sentences with Reinforcement Learning” Yogatama et al. ICLR 2017
[2] “Learning to Compose Task-Specific Tree Structures” Choi et al. AAAI 2018
[3] “Cooperative Learning of Disjoint Syntax and Semantics” Havrylov et al. NAACL 2019
[4] “ListOps: A Diagnostic Dataset for Latent Tree Learning” Nangia et al. NAACL 2018

Chart Parsers

Stack Augmented Recurrent Neural Networks

Latent Structure Learning Models

Reinforcement Learning or Biased Gradients

Chart Parsers [1,2]

- Can be comparatively expensive to run with longer sequences in practical situations.
- Have to recurse over the full sequence length keeping track of multiple paths of composition.

[1] “The Forest Convolutional Network: Compositional Distributional Semantics with a Neural Chart and without Binarization” Le at al. EMNLP 2015
[2] “Jointly learning sentence embeddings and syntax with unsupervised Tree-LSTMs” Maillard et al. Natural Language Engineering 2019

Stack Augmented Recurrent Neural Networks

Latent Structure Learning Models

Reinforcement Learning or Biased Gradients

Chart Parsers

Stack Augmented Recurrent Neural Networks [1,2,3]

- Have to recurse over the full sequence left to right.
- One of the most successful models (Ordered Memory[3]) uses a nested loop with an inner loop over its memory slots - adds overhead.

[1] "A fast unified model for parsing and sentence understanding." Bowman et al. ACL 2016
[2] "Learning to Compose Words into Sentences with Reinforcement Learning" Yogatama et al. ICLR 2017
[3] "Ordered Memory" Shen et al. NeurIPS 2019

Latent Structure Learning Models

Reinforcement Learning or Biased Gradients

Chart Parsers

Stack Augmented Recurrent Neural Networks

Proposed Approach:

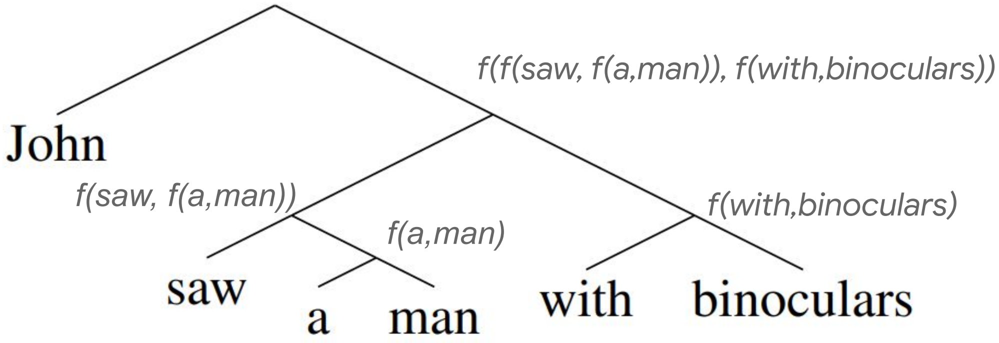
Continuous Recursive Neural Network (CRvNN)

- Backpropagation-friendly approximation of structure-inducing RvNN.
- Can learn to recurse over only the induced binary tree-depth by halting early.
- Can parallely compose or merge multiple child nodes (which makes it faster than Ordered Memory)

Continuous Recursive Neural Network (CRvNN)

- **We introduce a continuous relaxation to the structure** of a Recursive Neural Network to allow it to learn both the structure and the composition function through backpropagation.

f(John, f(f(saw, f(a,man)), f(with,binoculars)))



f() is a recursive composition function.

A New Look at RvNNs

- To make the shift from RvNNs to CRvNNs, we first re-formulate the original RvNNs in terms of **two rules** based of two sequences of binary values:
    - **Composition probabilities**
    - **Existential probabilities.**

A New Look at RvNNs

Given a sequence $x_{1:n}$ $(x_1, x_2, x_3, \ldots, x_n)$, we also maintain two sequences of binary probabilities - **composition probabilities** $c_{1:n}$ $(c_1, c_2, c_3, \ldots, c_n)$ and **existential probabilities** $e_{1:n}$ $(e_1, e_2, e_3, \ldots, e_n)$

A New Look at RvNNs

Given a sequence $x_{1:n}$ $(x_1, x_2, x_3, \dots, x_n)$, we also maintain two sequences of binary probabilities - **composition probabilities** $c_{1:n}$ $(c_1, c_2, c_3, \dots, c_n)$ and **existential probabilities** $e_{1:n}$ $(e_1, e_2, e_3, \dots, e_n)$

Iteration 1

$x_{1:n}$	x_1	x_2	x_3	x_4
$e_{1:n}$	1	1	1	1

Existential probabilities $e_i = 1$ means x_i is still "existing". $e_i = 0$ means x_i is treated as "non-existent"

A New Look at RvNNs

Iteration 1

$x_{1:n}$	x_1	x_2	x_3	x_4
$e_{1:n}$	1	1	1	1

→

Generate composition probabilities

$x_{1:n}$	x_1	x_2	x_3	x_4
$e_{1:n}$	1	1	1	1
$c_{1:n}$	0	1	0	0

A New Look at RvNNs

Iteration 1

$x_{1:n}$	x_1	x_2	x_3	x_4
$e_{1:n}$	1	1	1	1

Generate composition probabilities

$x_{1:n}$	x_1	x_2 →	x_3	x_4
$e_{1:n}$	1	1	1	1
$c_{1:n}$	0	**1**	0	0

A New Look at RvNNs

Iteration 1

$x_{1:n}$	x_1	x_2	x_3	x_4
$e_{1:n}$	1	1	1	1

Generate composition probabilities

$x_{1:n}$	x_1	$\mathbf{x_2}$ →	x_3	x_4
$e_{1:n}$	1	1	1	1
$c_{1:n}$	0	**1**	0	0

Update

$x_{1:n}$	x_1	---	$\mathbf{f(x_2,x_3)}$	x_4
$e_{1:n}$	1	0	1	1

A New Look at RvNNs

Iteration 2

$x_{1:n}$	x_1	---	$f(x_2,x_3)$	x_4
$e_{1:n}$	1	0	1	1

A New Look at RvNNs

Iteration 2

$x_{1:n}$	x_1	---	$f(x_2,x_3)$	x_4
$e_{1:n}$	1	0	1	1

→

Generate composition probabilities

$x_{1:n}$	x_1	---	$f(x_2,x_3)$	x_4
$e_{1:n}$	1	0	1	1
$c_{1:n}$	1	0	0	0

A New Look at RvNNs

Iteration 2

$x_{1:n}$	x_1	---	$f(x_2,x_3)$	x_4
$e_{1:n}$	1	0	1	1

Generate composition probabilities

$x_{1:n}$	$\mathbf{x_1}$	---	$f(x_2,x_3)$	x_4
$e_{1:n}$	1	0	1	1
$c_{1:n}$	**1**	0	0	0

A New Look at RvNNs

Iteration 2

$x_{1:n}$	x_1	---	$f(x_2,x_3)$	x_4
$e_{1:n}$	1	0	1	1

Generate composition probabilities

$x_{1:n}$	$\mathbf{x_1}$	---	$f(x_2,x_3)$	x_4
$e_{1:n}$	1	0	1	1
$c_{1:n}$	**1**	0	0	0

Update

$x_{1:n}$	---	---	$\mathbf{f(x_1,f(x_2,x_3))}$	x_4
$e_{1:n}$	0	0	1	1

A New Look at RvNNs

Iteration 3

$x_{1:n}$	---	---	$f(x_1,f(x_2,x_3))$	x_4
$e_{1:n}$	0	0	1	1

Generate composition probabilities

$x_{1:n}$	---	---	$\mathbf{f(x_1,f(x_2,x_3))}$ →	x_4
$e_{1:n}$	0	0	1	1
$c_{1:n}$	0	0	**1**	0

Update

$x_{1:n}$	---	---	---	$\mathbf{f(f(x_1,f(x_2,x_3)),x_4)}$
$e_{1:n}$	0	0	0	1

Mathematical Formalism

Update rules for iteration k:

$$x_i^{k+1} = left(c_i^k) \cdot \left(f(left(x_i^k), x_i^k)\right) + \left(1 - left(c_i^k)\right) \cdot x_i^k$$

$$e_i^{k+1} = e_i^k \cdot (1 - c_i^k)$$

$left(x_i^k)$ returns the immediately left item x_j^k after skipping over any value x_l^k with e_l^k as 0.

Towards Continuous Recursive Neural Networks

Update rules for iteration k:

$$x_i^{k+1} = left(c_i^k) \cdot \left(f(left(x_i^k), x_i^k)\right) + \left(1 - left(c_i^k)\right) \cdot x_i^k$$

$$e_i^{k+1} = e_i^k \cdot (1 - c_i^k)$$

- Use a model to predict $c^k_{i:n}$ to be in **[0,1]**; $e^k_{1:n}$ is also allowed to be in [0,1]
- Use a **soft attention-like neighbor retriever function left()** based on existential probabilities $e^k_{1:n}$

Dynamic Halt

When the complete tree is validly induced, the **final pattern of existential probabilities is the same** (0,0,0...,1).

Iteration 3

$x_{1:n}$	---	---	$f(x_1,f(x_2,x_3))$	x_4
$e_{1:n}$	0	0	1	1

Generate composition probabilities

$x_{1:n}$	---	---	$f(x_1,f(x_2,x_3))$ →	x_4
$e_{1:n}$	0	0	1	1
$c_{1:n}$	0	0	**1**	0

Update

$x_{1:n}$	---	---	---	$f(f(x_1,f(x_2,x_3)),x_4)$
$e_{1:n}$	0	0	0	1

Dynamic Halt

When the complete tree is validly induced, the **final pattern of existential probabilities is the same** (0,0,0...,1).

Iteration 3

$x_{1:n}$	---	---	$f(x_1,f(x_2,x_3))$	x_4
$e_{1:n}$	0	0	1	1

Generate composition probabilities

$x_{1:n}$	---	---	$\mathbf{f(x_1,f(x_2,x_3))} \rightarrow$	x_4
$e_{1:n}$	0	0	1	1
$c_{1:n}$	0	0	**1**	0

Update

$x_{1:n}$	---	---	---	$\mathbf{f(f(x_1,f(x_2,x_3)),x_4)}$
$e_{1:n}$	0	0	0	1

Halt early when existential probabilities are close to this pattern.

Experiments and Results

Datasets and Tasks

Synthetic Tasks

1. Logical Inference[1]
2. ListOps[2]

Natural Language Tasks

1. Natural Language Inference (SNLI[3], MultiNLI[4])
2. Sentiment Classification (SST2[5], SST5[5])

[1] Tree-structured composition in neural networks without tree-structured architectures" Bowman et al. International Conference on Cognitive Computation: Integrating Neural and Symbolic Approaches 2015
[2] "ListOps: A Diagnostic Dataset for Latent Tree Learning" Nangia et al. NAACL 2018
[3] "A large annotated corpus for learning natural language inference" Bowman et al. EMNLP 2015
[4] "A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference" Bowman et al. NAACL 2018
[5] "Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank" Socher et al. EMNLP 2013

Logical Inference[1]

$$(d (or\ f)) \sqsupset (f (and\ a))$$

$$(d (and (c (or\ d)))) \# (not\ f)$$

$$(not (d (or (f (or\ c))))) \sqsubset (not (c (and (not\ d))))$$

Need to predict the relationship: entailment ($\sqsubset$,$\sqsupset$) , independence (#), or something else?

Figure from: “The importance of being recurrent for modeling hierarchical structure” Tran et al. EMNLP 2018.
[1] “Tree-structured composition in neural networks without tree-structured architectures” Bowman et al. International Conference on Cognitive Computation: Integrating Neural and Symbolic Approaches 2015

Logical Inference

Model	Number of Operations					
	7	8	9	10	11	12
(Sentence representation models + ground truths)						
Tree-LSTM*	94	92	92	88	87	86
Tree-Cell*	98	96	96	95	93	92
Tree-RNN*	98	98	97	96	95	96
(Inter-sentence interaction models)						
Transformer*	51	52	51	51	51	48
Universal Transformer*	51	52	51	51	51	48
(Sentence representation models)						
LSTM*	88	84	80	78	71	69
RRNet*	84	81	78	74	72	71
ON-LSTM*	91	87	85	81	78	75
Ordered Memory*	98_0	97_4	96_5	94_8	93_5	92_{11}
(Our model)						
CRvNN	98_1	97_3	96_2	95_6	94_8	93_5

- Accuracy on Logical Inference dataset.
- Trained on data with less than 7 no. of operations.

98_1 means a standard deviation of +-0.1
*means that the results are reported from [1]

[1] “Ordered Memory” Shen et al. NeurIPS 2019

ListOps[1]

Multi-class (0-9) classification task

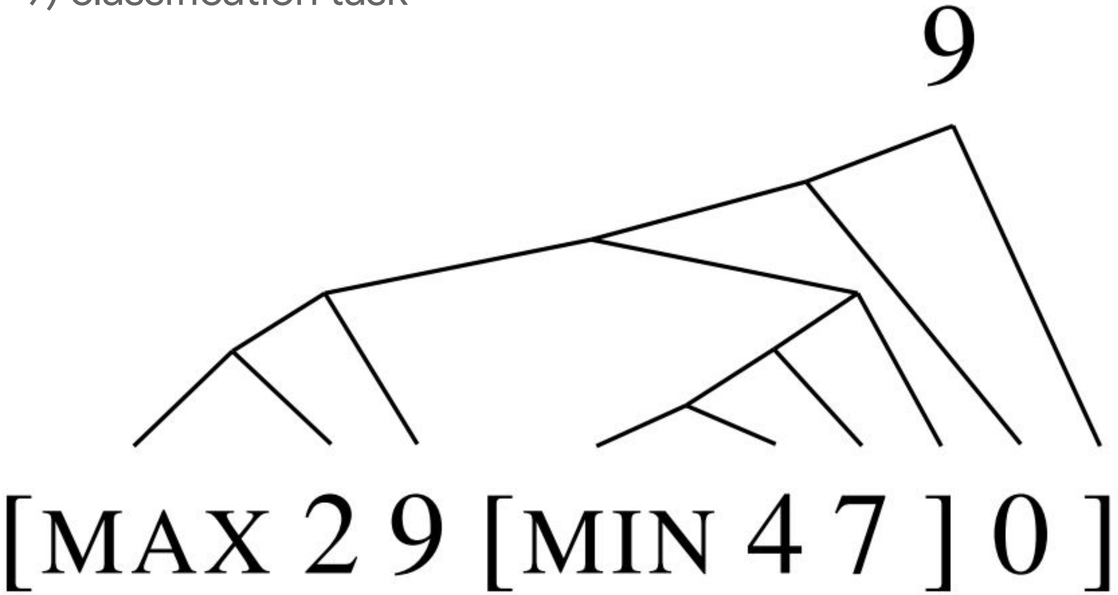


Figure from [1]

[1] “ListOps: A Diagnostic Dataset for Latent Tree Learning” Nangia et al. NAACL 2018

ListOps

Model	Accuracy
(Models with ground truth)	
Tree-LSTM$_{‡}$	98.7
(Models without ground truth)	
Transformer*	57.4±0.4
Universal Transformer*	71.5±7.8
LSTM †	71.5±1.5
RL-SPINN †	60.7±2.6
Gumbel-Tree LSTM †	57.6±2.9
(Havrylov et al., 2019) †	99.2 ±0.5
Ordered Memory*	99.97±0.014
(Our model)	
CRvNN	99.6±0.3

Results with * were taken from [1]. ‡ indicates that the results were taken from [2].† indicates that the results were taken from [3].

[1] "Ordered Memory" Shen et al. NeurIPS 2019
[2] "ListOps: A Diagnostic Dataset for Latent Tree Learning" Nangia et al. NAACL 2018
[3] "Cooperative Learning of Disjoint Syntax and Semantics" Havrylov et al. NAACL 2019

ListOps Length Extrapolation

Model	Sequence length ranges (ListOps)							
	200 – 300	300 – 400	400 – 500	500 – 600	600 – 700	700 – 800	800 – 900	900 – 1000
CRvNN	98.51±1.1	98.46±1.3	98.04±1.3	97.95±1.1	97.17±1.6	97.84±1.7	96.94±1.6	96.78±1.9

Natural Language Tasks

Model	SST2	SST5	SNLI	MNLI
RL-SPINN‡	—	—	82.3	67.4
Gumbel-Tree-LSTM††	90.7	53.7	85.6	—
Gumbel-Tree-LSTM‡	—	—	83.7	69.5
Gumbel-Tree-LSTM†	90.3_5	51.6_8	84.9_1	—
(Havrylov et al., 2019)†	90.2_2	51.5_4	85.1_2	70.7_3
Ordered Memory*	90.4	52.2	—	—
CRvNN	88.3_6	51.4_{13}	85.1_2	72.9_2

Accuracy on multiple natural language datasets. * indicates that the results were taken from [1] .† indicates that the results were taken from [2].‡ indicates that the results were taken from [3].†† indicates that the results were taken from [4]. $90_1 = 90 \pm 0.1$.

[1] "Ordered Memory" Shen et al. NeurIPS 2019
[2] "Cooperative Learning of Disjoint Syntax and Semantics" Havrylov et al. NAACL 2019
[3] "Do latent tree learning models identify meaningful structure in sentences?" Williams et al. TACL 2018
[4] "Learning to Compose Task-Specific Tree Structures" Choi et al. AAAI 2018

Acknowledgments

Thank You

Jishnu Ray Chowdhury (jraych2@uic.edu)

Cornelia Caragea (cornelia@uic.edu)

Github: https://github.com/JRC1995/Continuous-RvNN