Neural sequence modeling with tree structure

Jaemin Cho

2016.11.14.

Today we will cover

Tree-based models

- RNNs, SU-RNNs, MV-RNNs, RNTNs, Tree-LSTM, Tree2Seq, TBCNN ...

When are Tree Structures are effective?

- Novel word distribution model for tree-based models
  - Continuous Bag-of-Parsed Words (CBPW)

Two approaches in sequence modeling

- End-to-End
  - Input: sequence
  - Flexible
- Additional features
  - Input: sequence + additional features
  - (predetermined tags from preprocessing)
  - Better performance

Features in NLP

Words	Trump		also		likes		eating		sausages
POS-tag	Proper Noun		Adverb		Verb		Gerund		Noun
NER-tag	Name		-		-		-		-
Phrase-tag	Subject		Adverb		Verb		Object		Object
Head	likes		likes		ROOT		likes		eating
Word2vec	(0.1, 0.2)		(0.6, 0.5)		(0.9, 0.3)		(0.8, 0.7)		(0.7, 0.2)

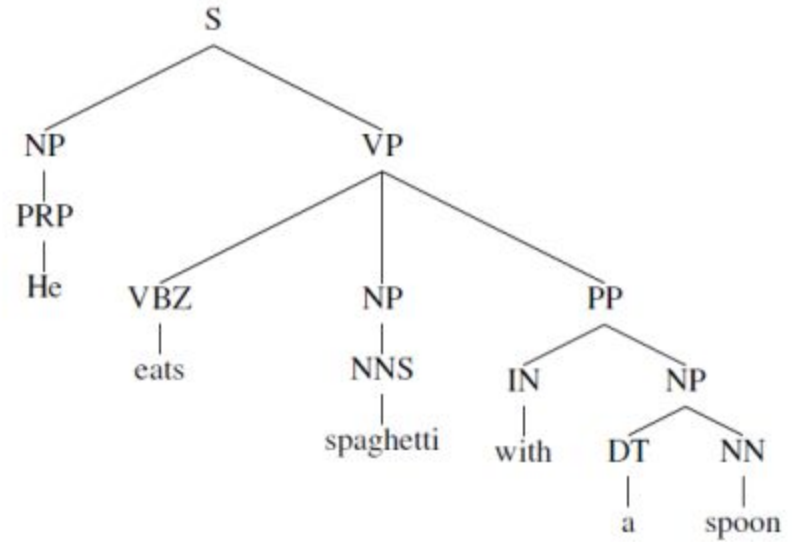
Why Tree structure?

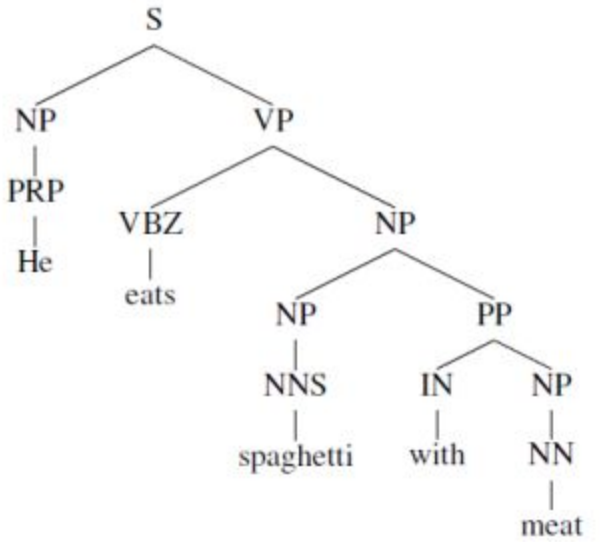
- What we actually want to understand
  - meaning of sentences and documents, not words!

- Simply adding, averaging word vectors left-to-right ignores sentence structure.
  - We need rules to combine them preserving words' meanings.

Are language constructed as tree?

- Debatable, but helpful in describing natural languages





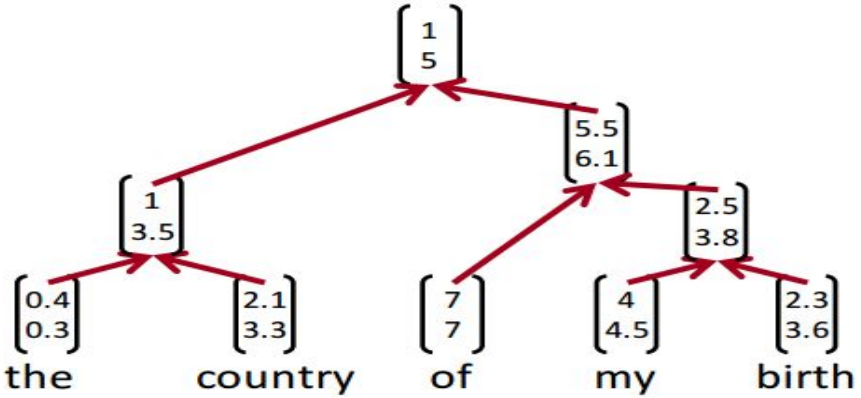
Since now, we assume to have parsers

- Parser will generate tree structure features from sentences

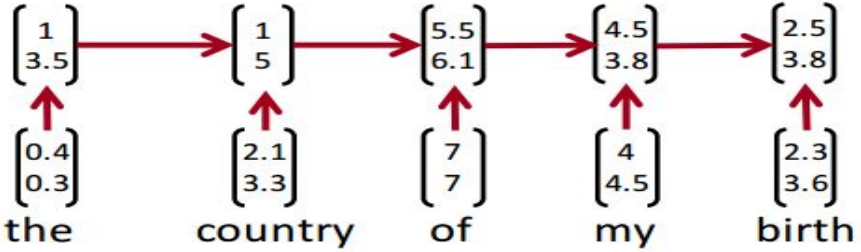
- For further NLP analysis,
  - first auto-parse all input sentences
  - And then use those features along with words.

Tree based models: RNNs (Recursive Neural Networks)

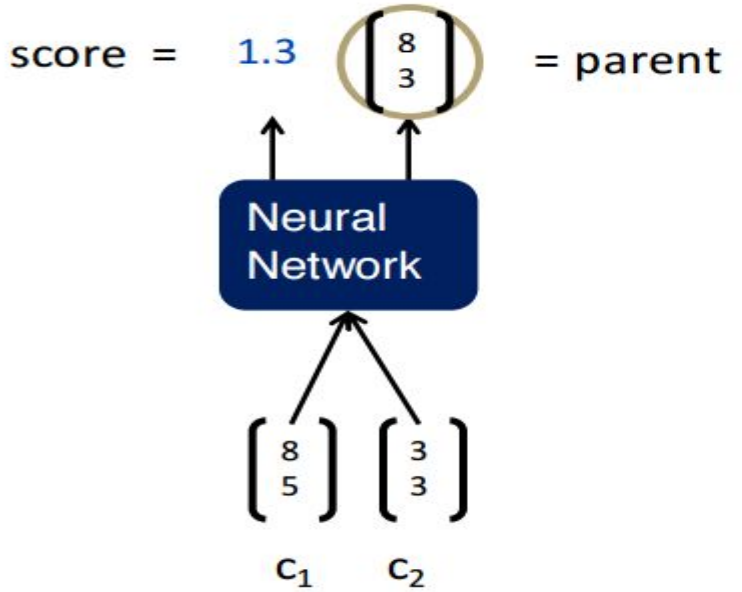
Recursive



Recurrent



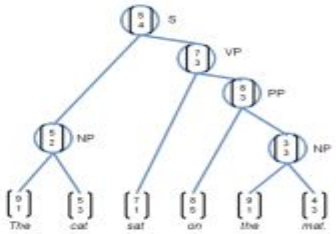
Tree based models: RNNs (Recursive Neural Networks)



score = $U^T p$

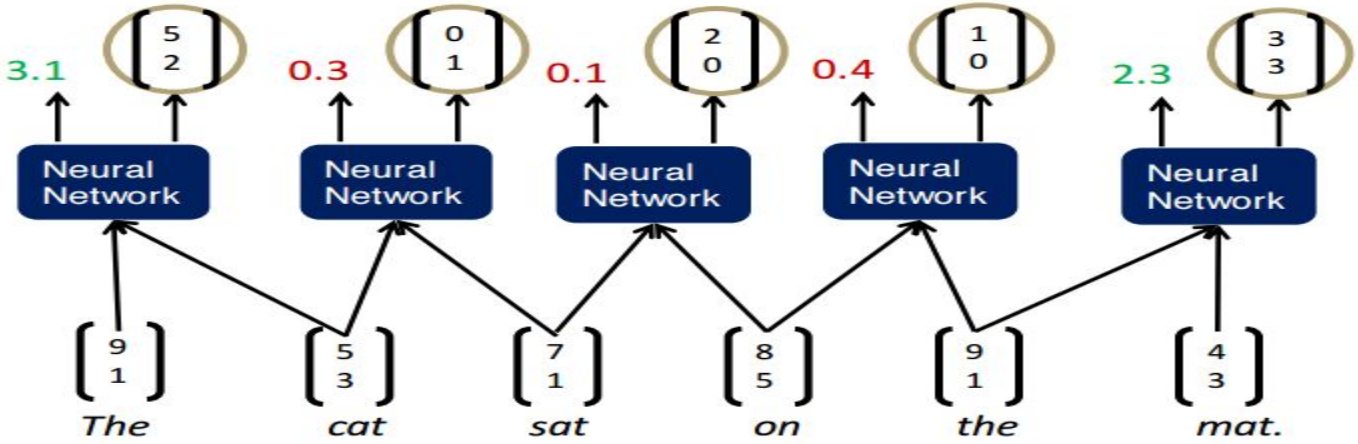
$$p = \tanh\left(W\begin{bmatrix} c_1 \\ c_2 \end{bmatrix} + b\right),$$

Same W parameters at all nodes of the tree



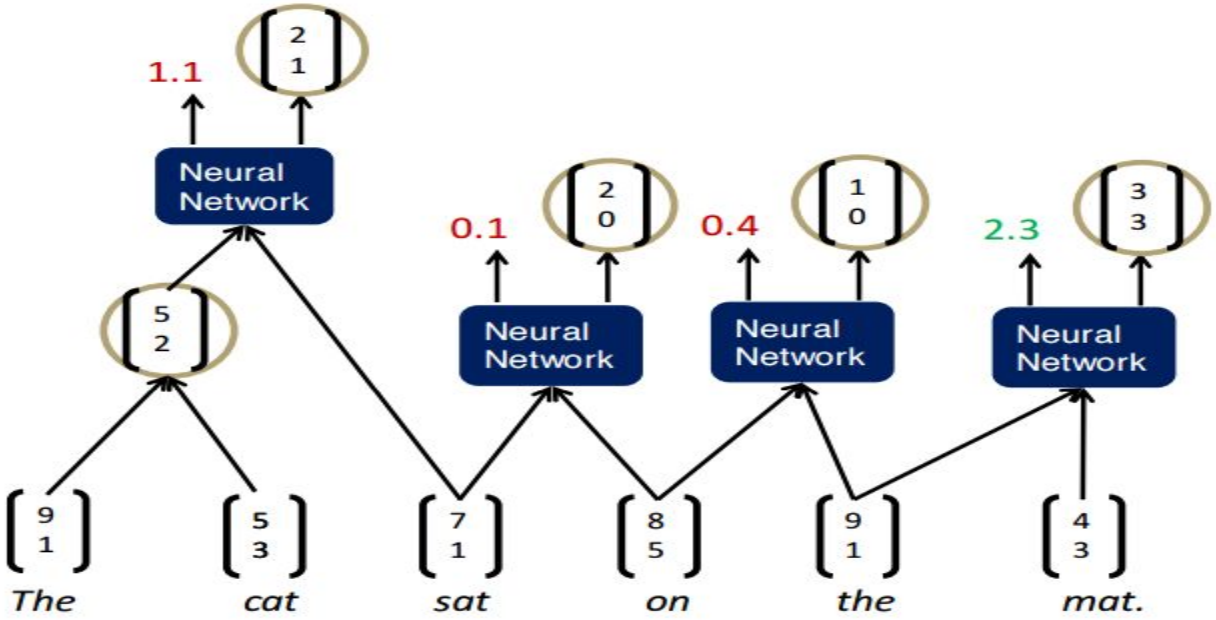
Tree based models: RNNs (Recursive Neural Networks)

- Parsing a sentence with an RNN



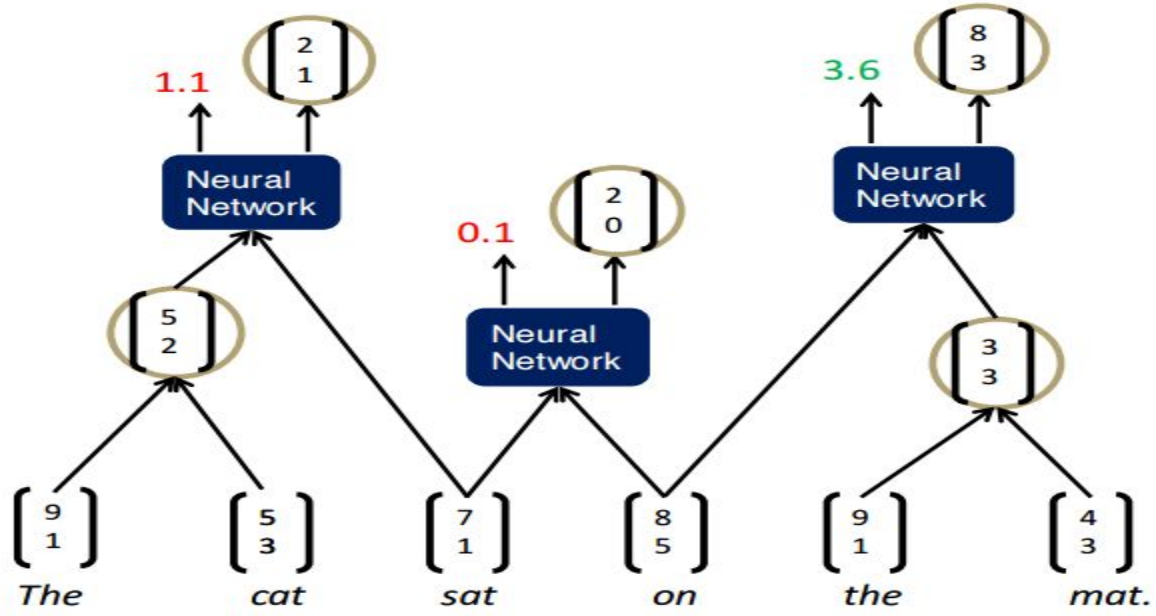
Tree based models: RNNs (Recursive Neural Networks)

- Parsing a sentence with an RNN



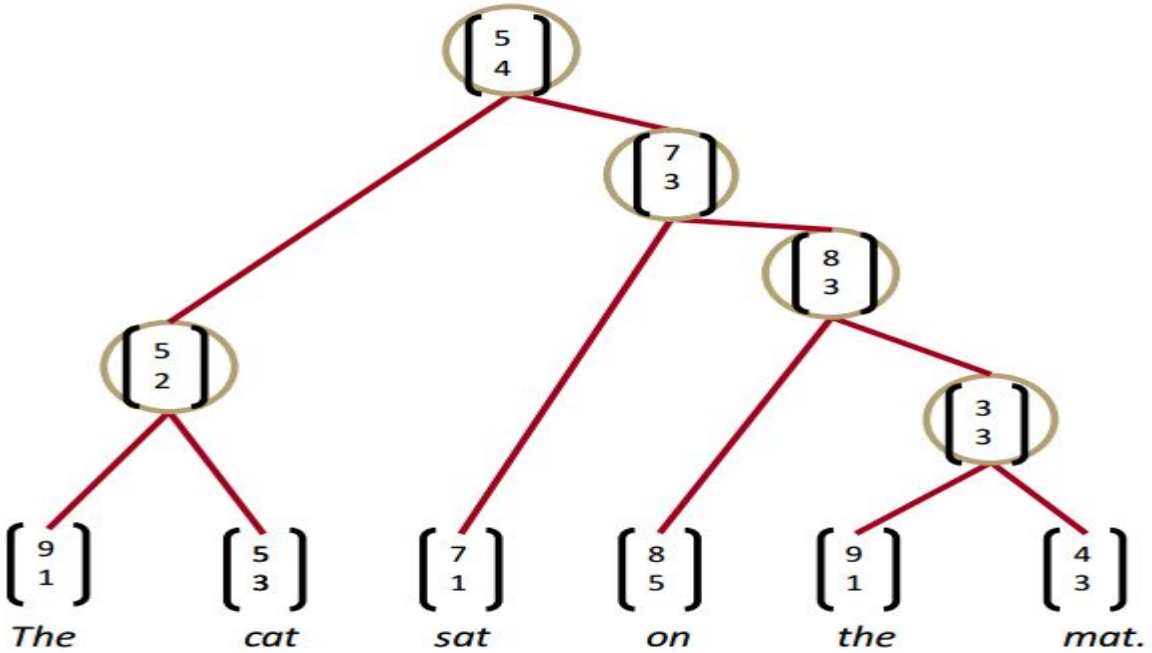
Tree based models: RNNs (Recursive Neural Networks)

- Parsing a sentence with an RNN

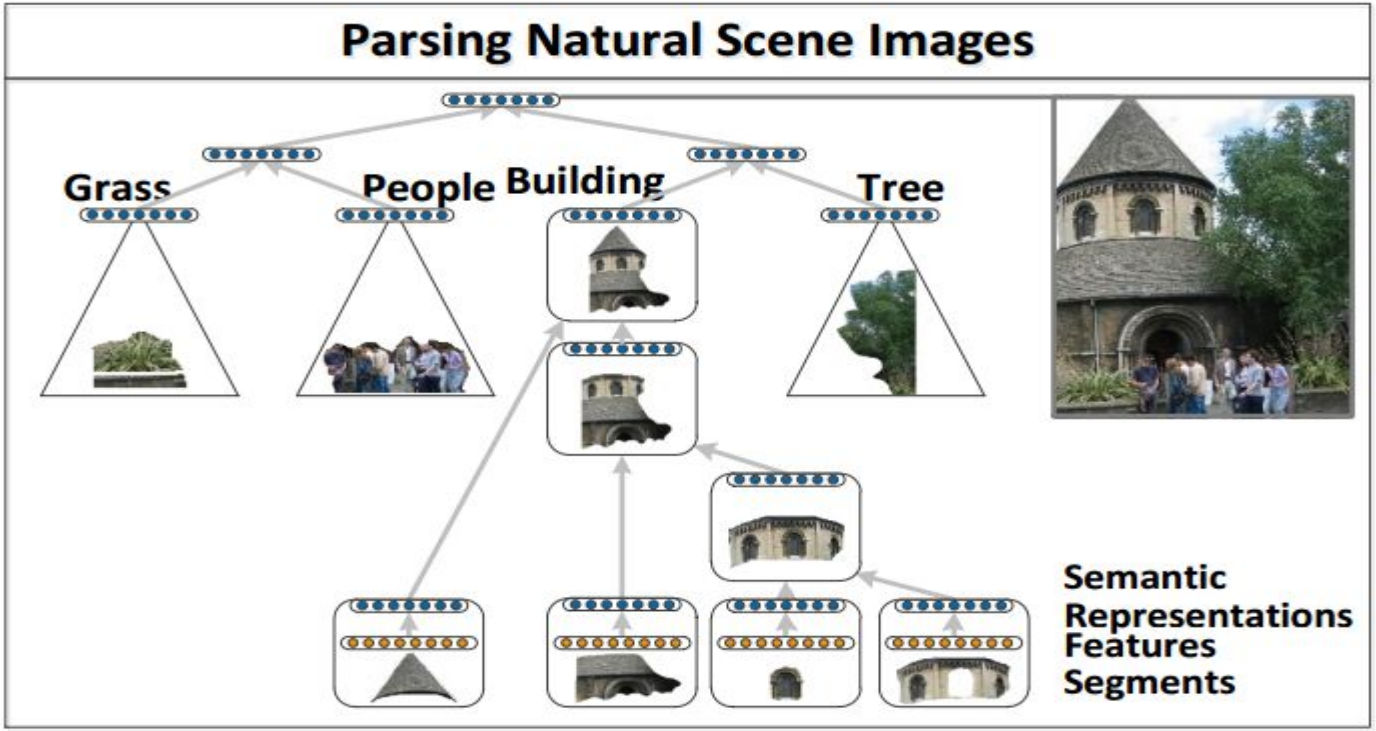


Tree based models: RNNs (Recursive Neural Networks)

- Parsing a sentence with an RNN

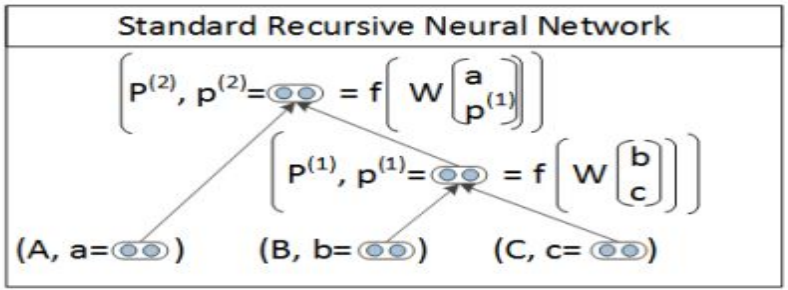


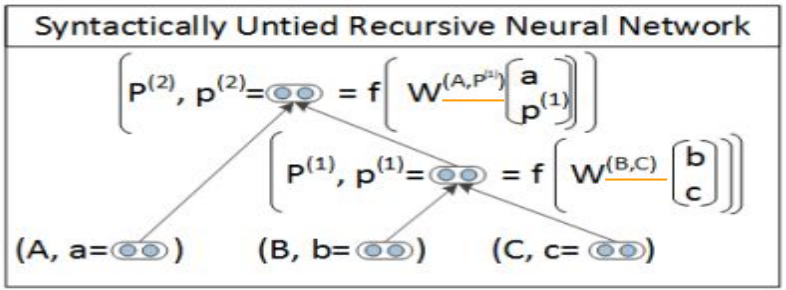
Tree based models: RNNs (Recursive Neural Networks)



Tree based models: SU-RNNs (Syntactically United RNNs)

- 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

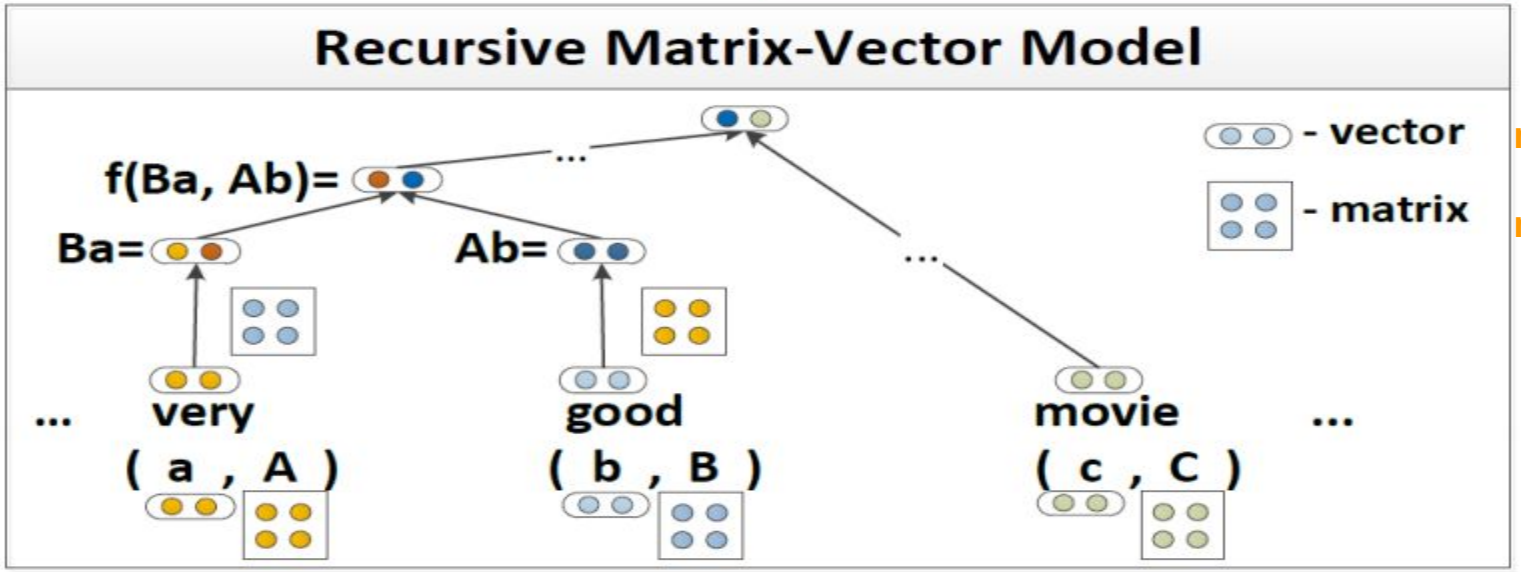




Tree based models: MV-RNNs (Matrix-Vector R

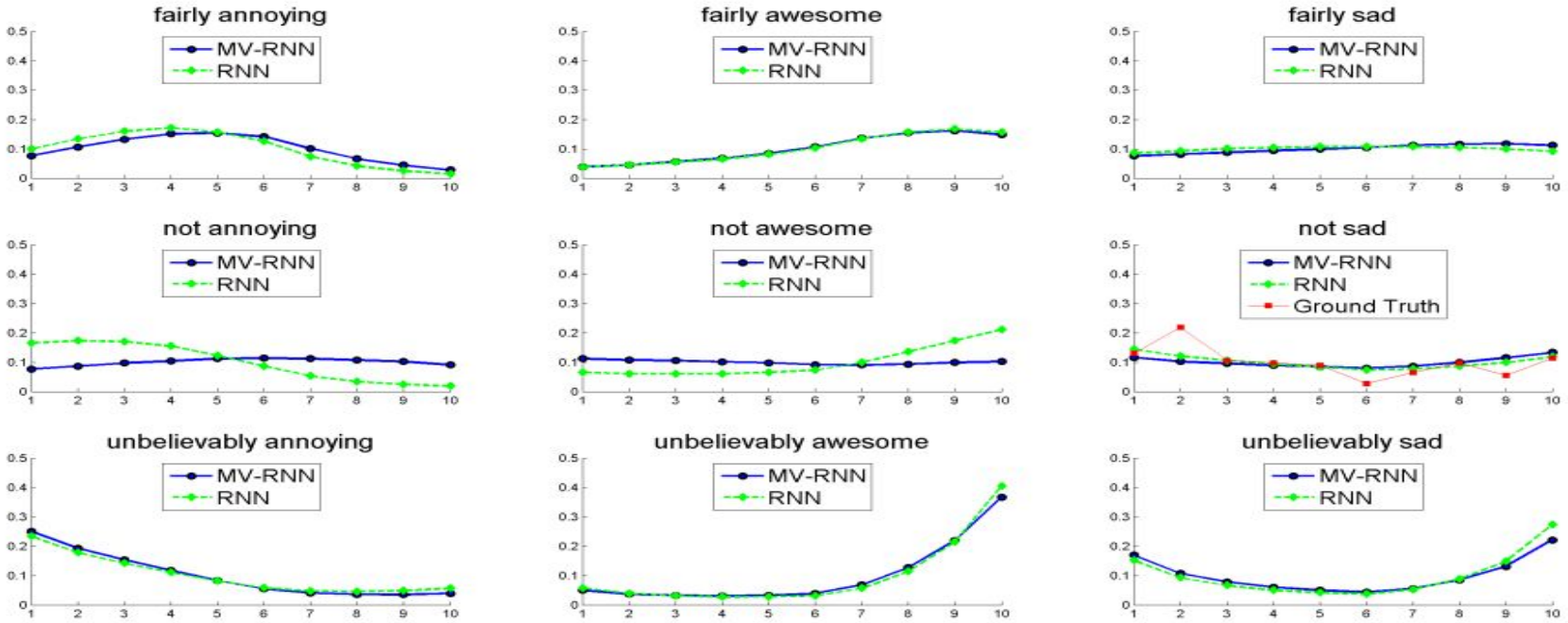
$$p = \tanh\left(W\begin{bmatrix}c_1\\c_2\end{bmatrix} + b\right)$$

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



Tree based models: MV-RNNs (Matrix-Vector

Good example for non-linearity in language

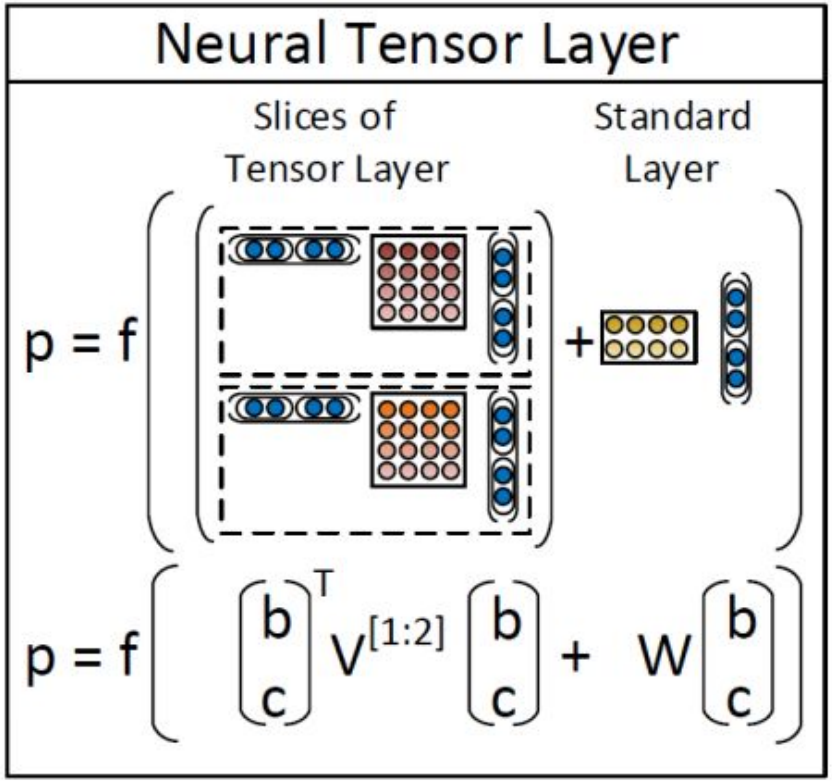


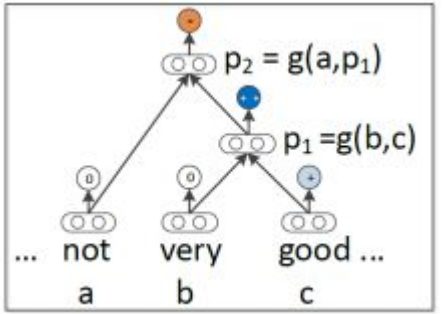
Tree based models: RNTNs (Recursive Neural Tensor Networks)

- MV-RNN still made three major mistakes
- Negated Positives
- Negated Negatives
- Contrastive Conjunctions (X but Y; X and Y have different sentiment)

- Need an even more expressive composition!

Tree based models: RNTNs (Recursive Neural Tensor Networks)





Tree based models: RNTNs (Recursive Neural Tensor Networks)

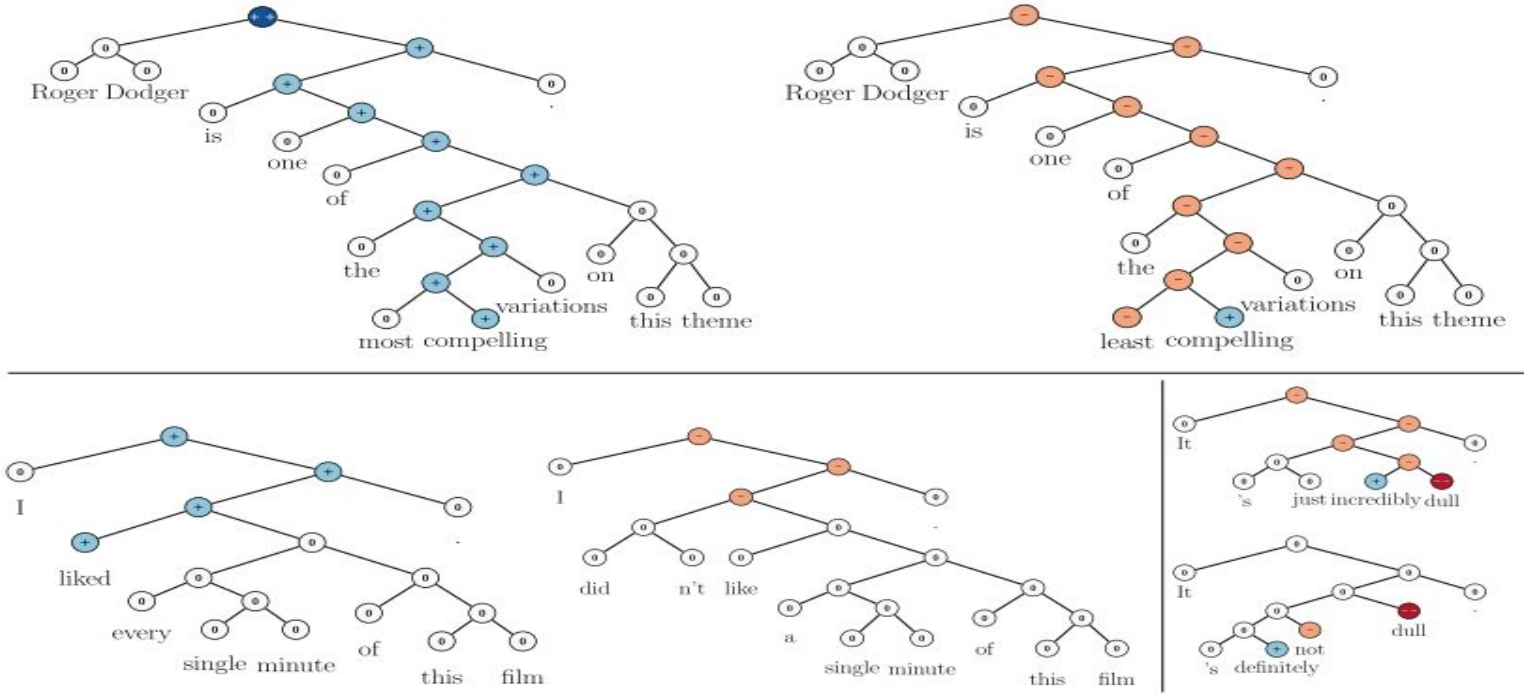


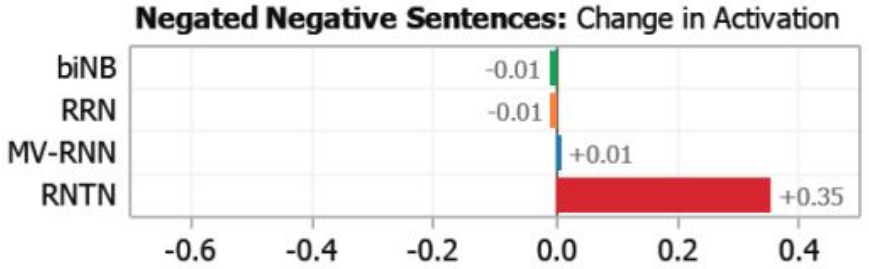
Figure 9: RNTN prediction of positive and negative (bottom right) sentences and their negation.

Tree based models: RNTNs (Recursive Neural Tensor Networks)

Model	Fine-grained		Positive/Negative	
	All	Root	All	Root
NB	67.2	41.0	82.6	81.8
SVM	64.3	40.7	84.6	79.4
BiNB	71.0	41.9	82.7	83.1
VecAvg	73.3	32.7	85.1	80.1
RNN	79.0	43.2	86.1	82.4
MV-RNN	78.7	44.4	86.8	82.9
RNTN	**80.7**	**45.7**	**87.6**	**85.4**

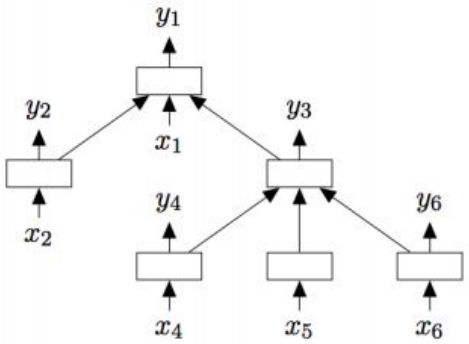
Table 1: Accuracy for fine grained (5-class) and binary predictions at the sentence level (root) and for all nodes.

Model	Accuracy	
	Negated Positive	Negated Negative
biNB	19.0	27.3
RNN	33.3	45.5
MV-RNN	52.4	54.6
RNTN	**71.4**	**81.8**



Tree based models: Tree-LSTM

Method	Fine-grained	Binary
RAE (Socher et al., 2013)	43.2	82.4
MV-RNN (Socher et al., 2013)	44.4	82.9
RNTN (Socher et al., 2013)	45.7	85.4
DCNN (Blunsom et al., 2014)	48.5	86.8
Paragraph-Vec (Le and Mikolov, 2014)	48.7	87.8
CNN-non-static (Kim, 2014)	48.0	87.2
CNN-multichannel (Kim, 2014)	47.4	**88.1**
DRNN (Irsoy and Cardie, 2014)	49.8	86.6
LSTM	45.8	86.7
Bidirectional LSTM	49.1	86.8
2-layer LSTM	47.5	85.5
2-layer Bidirectional LSTM	46.2	84.8
Constituency Tree LSTM (no tuning)	46.7	86.6
Constituency Tree LSTM	**50.6**	86.9



$$\tilde{h}_j = \sum_{k \in C(j)} h_k,$$

$$i_j = \sigma\left(W^{(i)} x_j + U^{(i)} \tilde{h}_j + b^{(i)}\right),$$

$$f_{jk} = \sigma\left(W^{(f)} x_j + U^{(f)} h_k + b^{(f)}\right),$$

$$o_j = \sigma\left(W^{(o)} x_j + U^{(o)} \tilde{h}_j + b^{(o)}\right),$$

$$u_j = \tanh\left(W^{(u)} x_j + U^{(u)} \tilde{h}_j + b^{(u)}\right),$$

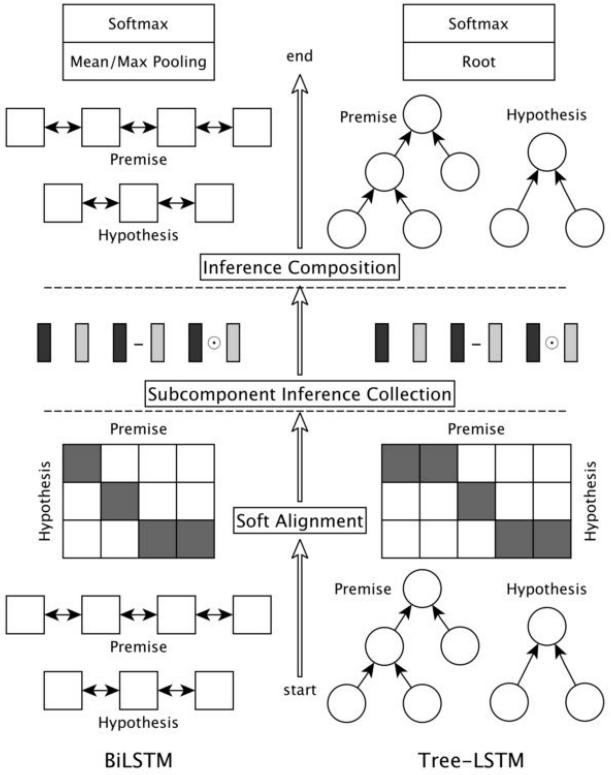
$$c_j = i_j \odot u_j + \sum_{k \in C(j)} f_{jk} \odot c_k,$$

$$h_j = o_j \odot \tanh(c_j),$$

Tree based models: Ensemble: Tree&Bi-LSTM

Table 1: Performance (accuracy) of models on the benchmark data SNLI. Our final model achieves the accuracy of 88.3%, the best result seen so far on SNLI, while our enhanced sequential encoding model attains an accuracy of 87.7%, which also outperform the previous models. The numbers of parameters reported here, as in previous work, do not include word embeddings vectors.

Model	#Para.	Train	Test
(1) Handcrafted features [Bowman et al., 2015]	-	99.7	78.2
(2) 300D LSTM encoders [Bowman et al., 2016]	3.0M	83.9	80.6
(3) 1024D pretrained GRU encoders [Vendrov et al., 2015]	15M	98.8	81.4
(4) 300D tree-based CNN encoders [Mou et al., 2016]	3.5M	83.3	82.1
(5) 300D SPINN-PI encoders [Bowman et al., 2016]	3.7M	89.2	83.2
(6) 600D BiLSTM intra-attention encoders [Liu et al., 2016]	2.8M	84.5	84.2
(7) 300D NSE encoders [Munkhdalai and Yu, 2016a]	3.0M	86.2	84.6
(8) 100D LSTM with attention [Rocktäschel et al., 2015]	250k	85.3	83.5
(9) 300D mLSTM [Wang and Jiang, 2015]	1.9M	92.0	86.1
(10) 450D LSTMN with deep attention fusion [Cheng et al., 2016]	3.4M	88.5	86.3
(11) 200D decomposable attention model [Parikh et al., 2016]	380K	89.5	86.3
(12) Intra-sentence attention + (11) [Parikh et al., 2016]	580K	90.5	86.8
(13) 300D NTI-SLSTM-LSTM [Munkhdalai and Yu, 2016b]	3.2M	88.5	87.3
(14) 600D EBIM	8.6M	92.9	**87.7**
(15) 600D EBIM + 300D Syntactic tree-LSTM	12M	93.0	**88.3**



Tree based models: Tree2Seq

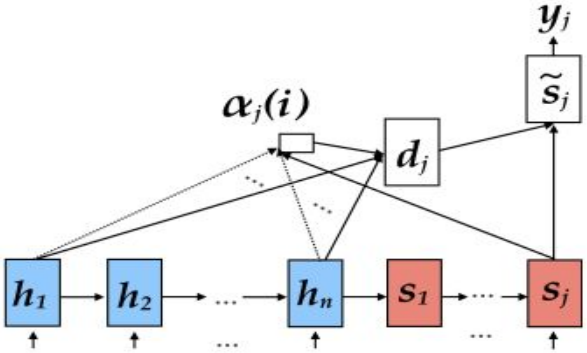


Figure 2: Attentional Encoder-Decoder model.

$$
\begin{aligned}
\boldsymbol{i}_t &= \sigma(\boldsymbol{W}^{(i)}\boldsymbol{x}_t + \boldsymbol{U}^{(i)}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{(i)}),\\
\boldsymbol{f}_t &= \sigma(\boldsymbol{W}^{(f)}\boldsymbol{x}_t + \boldsymbol{U}^{(f)}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{(f)}),\\
\boldsymbol{o}_t &= \sigma(\boldsymbol{W}^{(o)}\boldsymbol{x}_t + \boldsymbol{U}^{(o)}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{(o)}),\\
\tilde{\boldsymbol{c}}_t &= \tanh(\boldsymbol{W}^{(\tilde{c})}\boldsymbol{x}_t + \boldsymbol{U}^{(\tilde{c})}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{(\tilde{c})}),\\
\boldsymbol{c}_t &= \boldsymbol{i}_t \odot \tilde{\boldsymbol{c}}_t + \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1},\\
\boldsymbol{h}_t &= \boldsymbol{o}_t \odot \tanh(\boldsymbol{c}_t),
\end{aligned}
$$

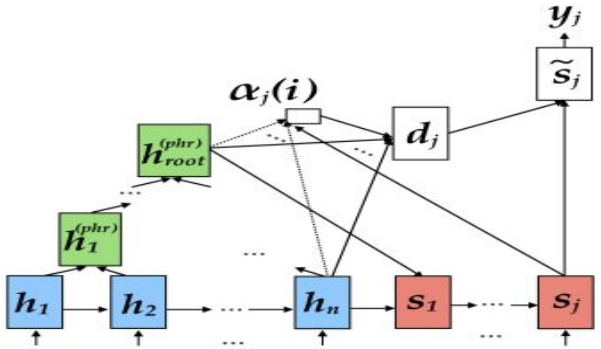


Figure 3: Proposed model: Tree-to-sequence attentional NMT model.

$$
\begin{aligned}
\boldsymbol{i}_k &= \sigma(\boldsymbol{U}_l^{(i)}\boldsymbol{h}_k^l + \boldsymbol{U}_r^{(i)}\boldsymbol{h}_k^r + \boldsymbol{b}^{(i)}),\\
\boldsymbol{f}_k^l &= \sigma(\boldsymbol{U}_l^{(f_l)}\boldsymbol{h}_k^l + \boldsymbol{U}_r^{(f_l)}\boldsymbol{h}_k^r + \boldsymbol{b}^{(f_l)}),\\
\boldsymbol{f}_k^r &= \sigma(\boldsymbol{U}_l^{(f_r)}\boldsymbol{h}_k^l + \boldsymbol{U}_r^{(f_r)}\boldsymbol{h}_k^r + \boldsymbol{b}^{(f_r)}),\\
\boldsymbol{o}_k &= \sigma(\boldsymbol{U}_l^{(o)}\boldsymbol{h}_k^l + \boldsymbol{U}_r^{(o)}\boldsymbol{h}_k^r + \boldsymbol{b}^{(o)}),\\
\tilde{\boldsymbol{c}}_k &= \tanh(\boldsymbol{U}_l^{(\tilde{c})}\boldsymbol{h}_k^l + \boldsymbol{U}_r^{(\tilde{c})}\boldsymbol{h}_k^r + \boldsymbol{b}^{(\tilde{c})}),\\
\boldsymbol{c}_k^{(phr)} &= \boldsymbol{i}_k \odot \tilde{\boldsymbol{c}}_k + \boldsymbol{f}_k^l \odot \boldsymbol{c}_k^l + \boldsymbol{f}_k^r \odot \boldsymbol{c}_k^r,\\
\boldsymbol{h}_k^{(phr)} &= \boldsymbol{o}_k \odot \tanh(\boldsymbol{c}_k^{(phr)}), \qquad (1
\end{aligned}
$$

Tree based models: Tree2Seq

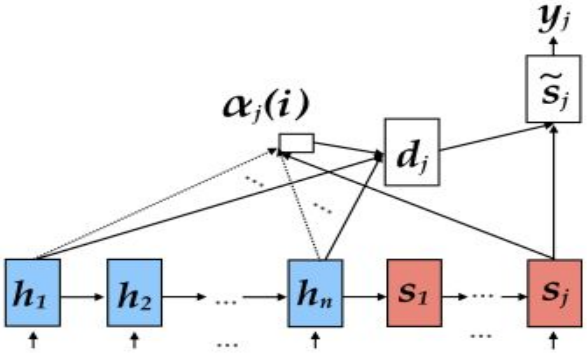


Figure 2: Attentional Encoder-Decoder model.

$$\alpha_j(i) = \frac{\exp(\boldsymbol{h}_i \cdot \boldsymbol{s}_j)}{\sum_{k=1}^{n} \exp(\boldsymbol{h}_k \cdot \boldsymbol{s}_j)} \qquad \boldsymbol{d}_j = \sum_{i=1}^{n} \alpha_j(i)\boldsymbol{h}_i$$

$$\tilde{\boldsymbol{s}}_j = \tanh(\boldsymbol{W}_d[\boldsymbol{s}_j; \boldsymbol{d}_j] + \boldsymbol{b}_d), \quad p(y_j|\boldsymbol{y}_{<j}, \boldsymbol{x}) = \text{softmax}(\boldsymbol{W}_s\tilde{\boldsymbol{s}}_j + \boldsymbol{b}_s)$$

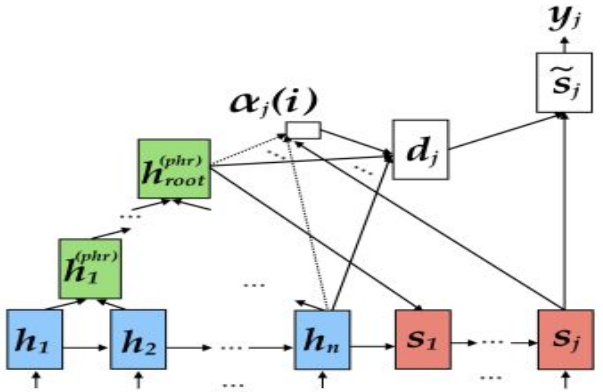


Figure 3: Proposed model: Tree-to-sequence attentional NMT model.

$$\tilde{c}_k = \tanh(U_l^{(\tilde{c})}h_k^l + U_r^{(\tilde{c})}h_k^r + b^{(\tilde{c})}) \qquad \boldsymbol{d}_j = \sum_{i=1}^{n} \alpha_j(i)\boldsymbol{h}_i + \sum_{i=n+1}^{2n-1} \alpha_j(i)\boldsymbol{h}_i^{(phr)}$$

$$c_k^{(phr)} = i_k \odot \tilde{c}_k + f_k^l \odot c_k^l + f_k^r \odot c_k^r$$
$$h_k^{(phr)} = o_k \odot \tanh(c_k^{(phr)}), \qquad \boldsymbol{s}_j = f_{dec}(y_{j-1}, [\boldsymbol{s}_{j-1}; \tilde{\boldsymbol{s}}_{j-1}])$$

Tree based models: Tree2Seq

Model	RIBES	BLEU
Proposed model ($d = 512$)	81.46	34.36
Proposed model ($d = 768$)	81.89	34.78
Proposed model ($d = 1024$)	81.58	34.87
Ensemble of the above three models	**82.45**	36.95
ANMT with LSTMs (Zhu, 2015)	79.70	32.19
+ Ensemble, *unk* replacement	80.27	34.19
+ System combination, 3 pre-reordered ensembles	80.91	36.21
ANMT with GRUs (Lee et al., 2015) + character-based decoding, Begin/Inside representation	81.15	35.75
PB baseline	69.19	29.80
HPB baseline	74.70	32.56
T2S baseline	75.80	33.44
T2S model (Neubig and Duh, 2014)	79.65	36.58
+ ANMT Rerank (Neubig et al., 2015)	81.38	38.17

Table 6: Evaluation results on the test data.

Tree based models: TBCNN (Tree-based CNN)

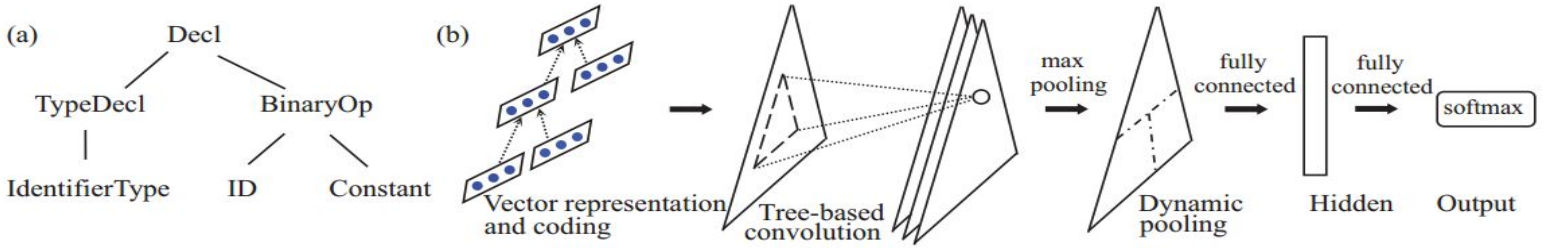
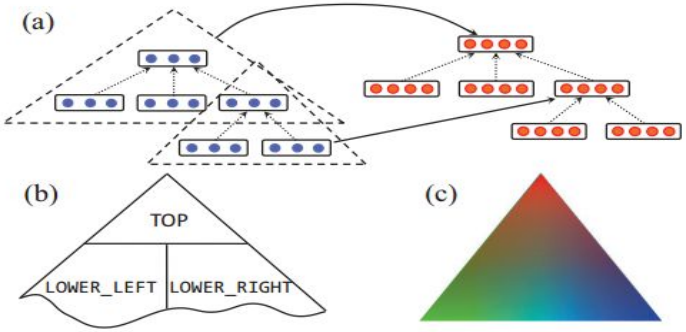


Figure 1: (a) Illustration of an AST, corresponding to the C code snippet "int a=b+3;" It should be notice that our model takes as input the entire AST of a program, which is typically much larger. (b) The architecture of the Tree-Based Convolutional Neural Network (TBCNN). The main components in our model include vector representation and coding, tree-based convolution and dynamic pooling; then a fully-connected hidden layer and an output layer (softmax) are added.



Recursive models are better than Recurrent model?

- When Are Tree Structures Necessary for Deep Learning of Representations? (2015)
- investigate 4 tasks:

    (1) sentiment classification at the sentence level and phrase level

    (2) matching questions to answer phrases

    (3) discourse parsing

    (4) semantic relation extraction.

Recursive models are better than Recurrent model?

- Almost equivalent
- Components from Recurrent models are able to embed important information, despite the fact their components are just sentence fragments and hence usually not linguistically meaningful components in the way that parse tree constituents are.
- A Simple approximation to tree-based models can improve recurrent models to achieve almost equivalent performance
    - Break long sentences, work on phrases separately, join them together
- When Recursive shines: sentiment analysis in long sentences
- Note: benchmarks on simple algorithms on papers, rich features can always improve performance

References

- Stanford CS 224D: Deep Learning for NLP (2016)
- Parsing Natural Scenes and Natural Language with Recursive Neural Networks (2011)
- Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank (2013)
- Convolutional Neural Networks over Tree Structures for Programming Language Processing (2014)
- Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks (2015)
- Enhancing and Combining Sequential and Tree LSTM for Natural Language Inference (2016)
- Tree-to-Sequence Attentional Neural Machine Translation (2016)
- When Are Tree Structures Necessary for Deep Learning of Representations? (2015)