# Fast Tree-Structured RNTN

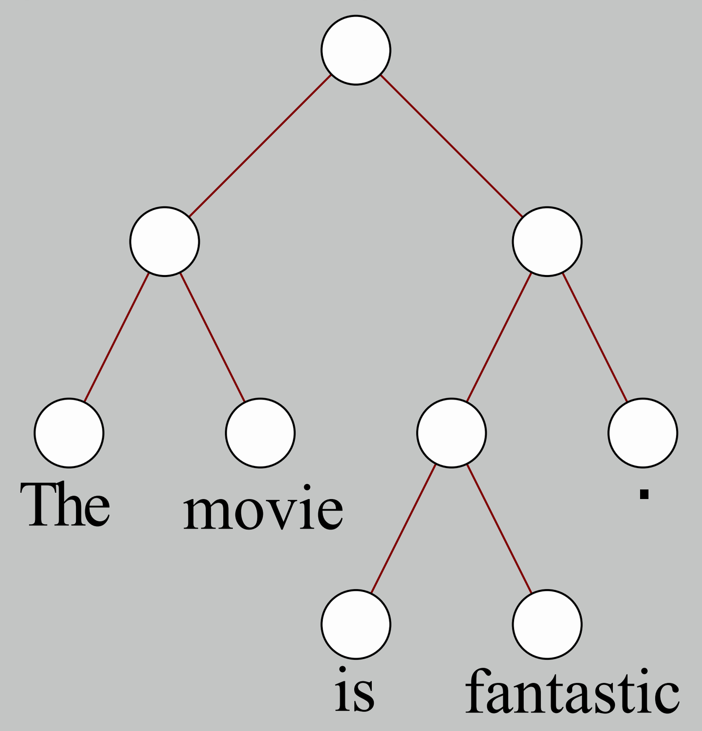
Anand Avati, Nai-Chia Chen
Stanford University

## Summary


- RNTN has been previously successfully applied to sentiment analysis. However, its training process is time-consuming.
- Our objective is to speed up the training of RNTNs with mini-batch gradient descent, which involves many matrix-vector multiplications.
- The existing code performs forward and backward propagation one example at a time, and the matrix-vector multiplication is executed one node at a time.
- We observe that the weight matrix is shared across training examples and across nodes, making it possible to combine many matrix-vector multiplications into one large matrix-matrix multiplication.
- We have implemented our idea. As a result, the training time has been significantly reduced.

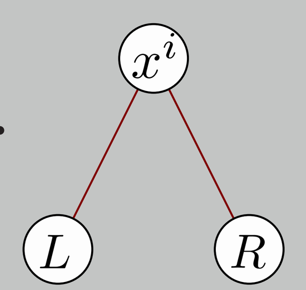

## Recursive Neural Tensor Network (RNTN)

- Each training example has its own parse-tree structure.




- Parameters $\theta = (L, W, W_s, V)$.
  $L$: word embedding matrix. $W$: weight matrix $W$.
  $W_s$: classification matrix. $V$: weight tensor.
- Node vectors are computed starting from the leaf nodes.
- Representation at node $i$:

$$x^i = \tanh\left(\begin{pmatrix} x_L \\ x_R \end{pmatrix}^T V \begin{pmatrix} x_L \\ x_R \end{pmatrix} + W \begin{pmatrix} x_L \\ x_R \end{pmatrix}\right) \in \mathbb{R}^d.$$



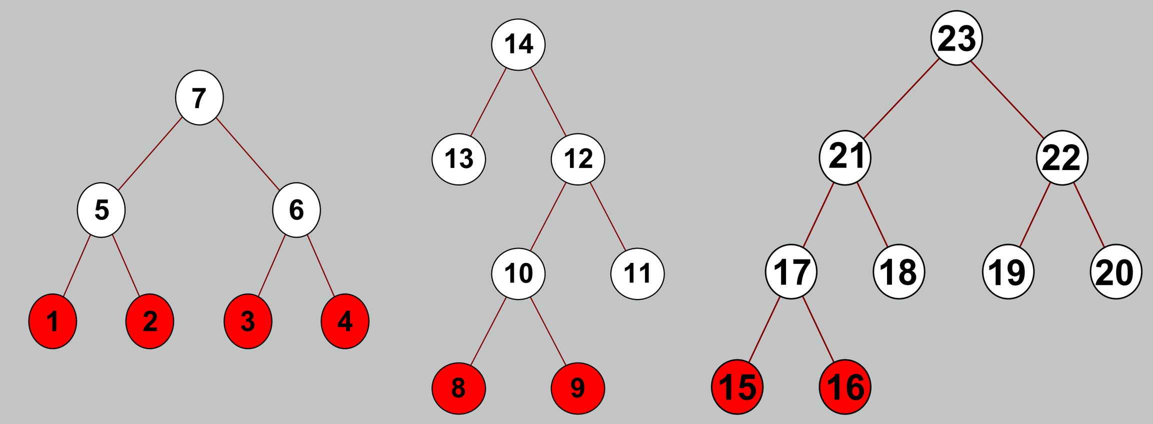

- Soft classification at node $x^i$ :

$$y^i = \mathrm{softmax}(W_s x^i).$$

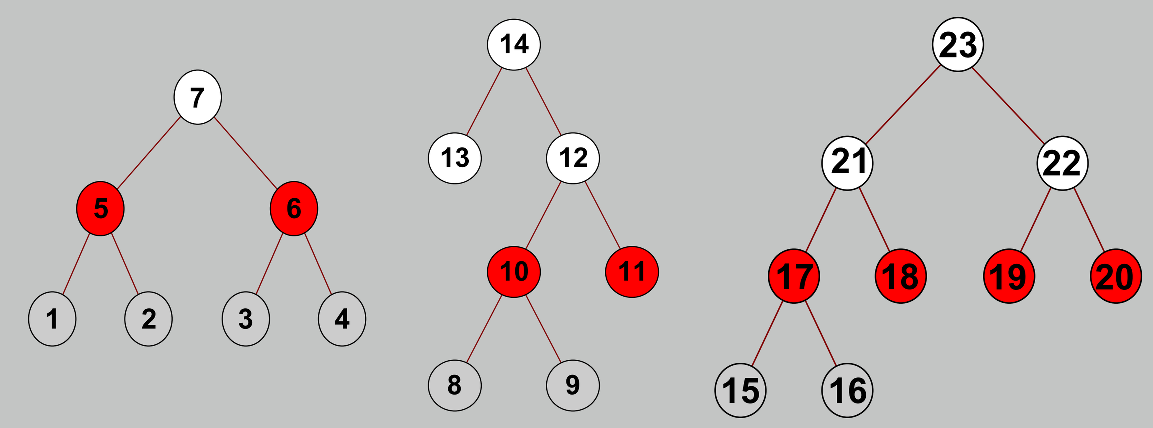
- Each node $x^i$ has a target vector $t^i$.
- Error function $E(\theta) = -\sum_i < t^i, \log y^i > + \lambda ||\theta||^2$.
- Training:
  - Data set: Stanford Sentiment Treebank
  - Batched Adaptive Gradient Descent (AdaGrad)
  - Existing code: CoreNLP

## Forward Propagation

- Grouping Nodes Across Trees (assume batch size=3)
- Step 1:



- Step 2:



- Step 3: Traverse up until all nodes are visited.

## Grouping $k$ pairs of nodes

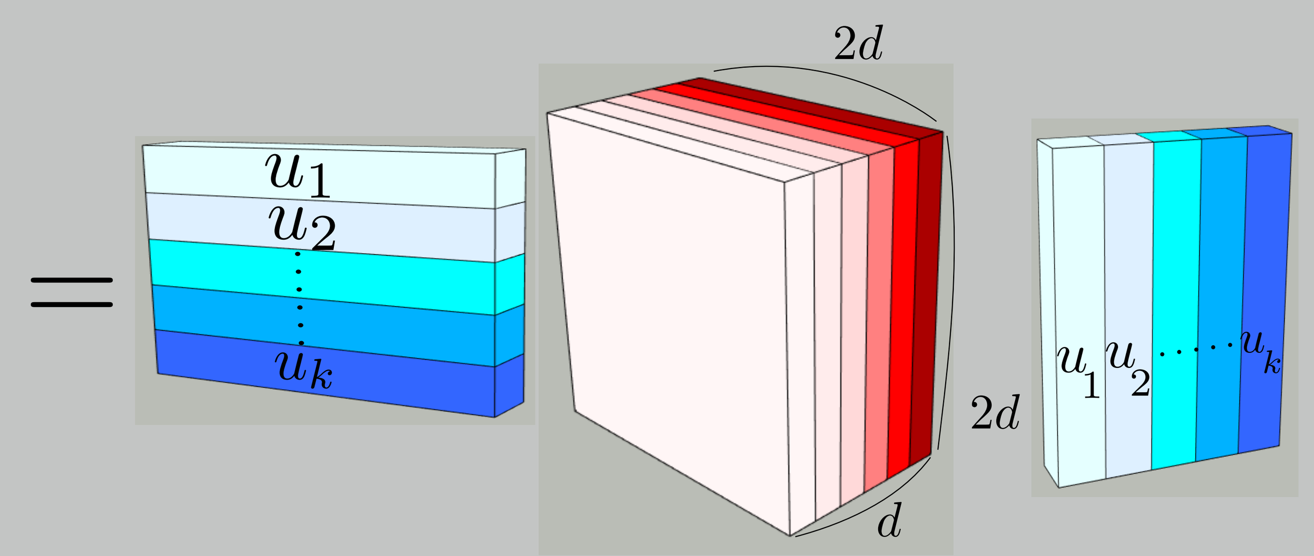
Concatenation vectors: let $u_i = \begin{bmatrix} x_L^i \\ x_R^i \end{bmatrix} \in \mathbb{R}^{2d}, i = 1, \cdots, k$

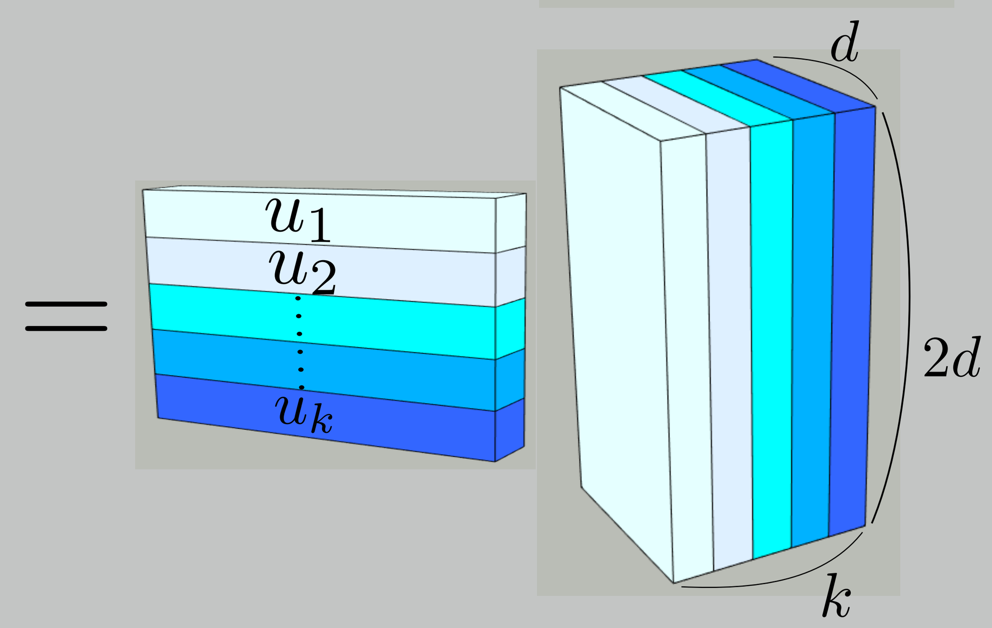
- Grouping matrix-vector multiplication: $W \in \mathbb{R}^{2d \times d}$

$$\left[Wu_1 \; Wu_2 \; \cdots \; Wu_k\right] = W \left[u_1 \; u_2 \; \cdots \; u_k\right].$$

- Grouping tensor-vector operation: $V \in \mathbb{R}^{2d \times 2d \times d}$

$$\left[u_1^T V u_1 \; u_2^T V u_2 \; \cdots \; u_k^T V u_k\right] \quad \in \mathbb{R}^{d \times k}$$




## Backward Propagation

- Formula for errors $\delta^i$ at node $i$.
  s: softmax error com: complete incoming error

$$\delta^{i,s} = W_s^T (y^i - t^i)$$

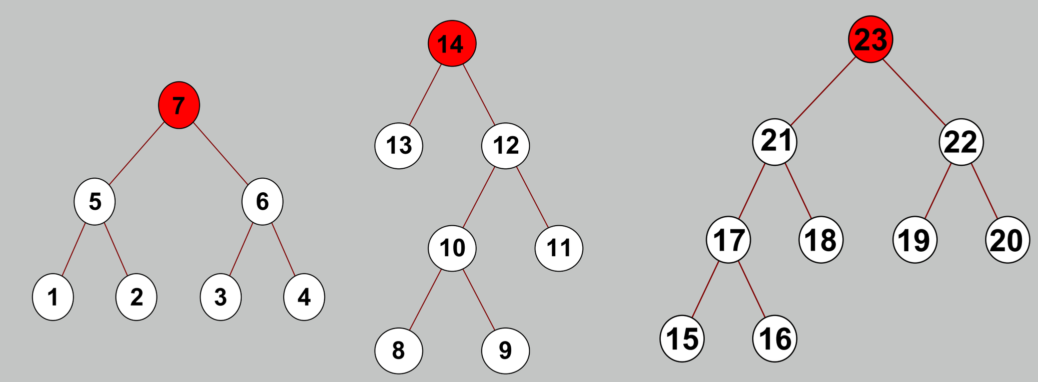
$$\delta^{i,com} = \begin{cases} \delta^{i,s} & \text{if } x^i \text{ is the root} \\ \delta^{p(i),down}[1:d] & \text{if } x^i \text{ is the left child of } x^{p(i)} \\ \delta^{p(i),down}[d+1:2d] & \text{if } x^i \text{ is the right child of } x^{p(i)} \end{cases}$$

$$\delta^{i,down} = (W^T \delta^{i,com} + S^i) \circ f'(x^i),$$

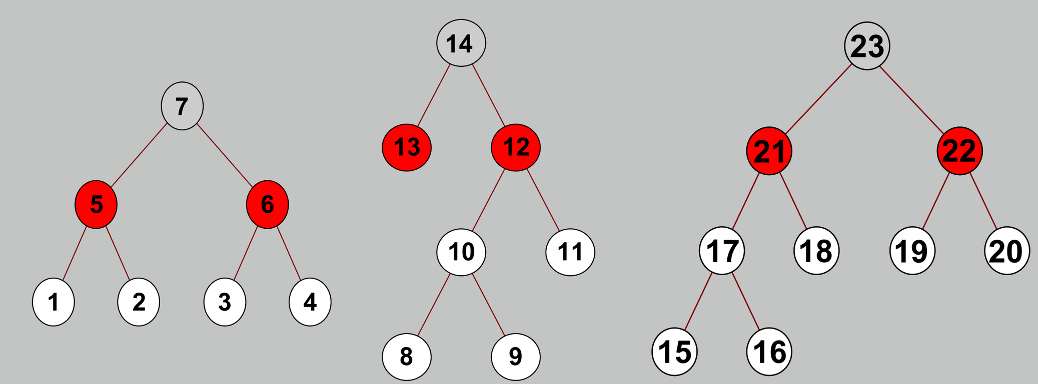
where $\circ$ : Hadamard product, $f'(x) = 1 - x^2$.

$$S^i = \sum_{\ell=1}^{d} \delta_\ell^{i,com} \left(V^{[\ell]} + (V^{[\ell]})^T\right) \begin{bmatrix} x_L^i \\ x_R^i \end{bmatrix}.$$

- Grouping Nodes Across Trees
- Step 1:



- Step 2:



- Step 3: Traverse down until all nodes are visited.

## Results

- Word Vector size = 50.

| Batch Size | Training Time Per Epoch (sec) Before | After | Reduce |
|---|---|---|---|
| 25 | 1529 | 1012 | 33.8% |
| 50 | 1532 | 975 | 36.4% |
| 100 | 1567 | 9650 | 38.4% |

- Batch size = 50.

| Word Vector Size | Training Time Per Epoch (sec) Before | After | Reduce |
|---|---|---|---|
| 25 | 191 | 136 | 28.8% |
| 50 | 1533 | 975 | 36.4% |
| 100 | 12711 | 7828 | 38.4% |

- Our code is at `https://github.com/avati/CoreNLP/tree/treernn`