Improving Constituent Representation with Hypertree Neural Networks

Hao Zhou [1], Gongshen Liu [1], Kewei Tu [2]

[1] School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University

[2] School of Information Science and Technology, ShanghaiTech University;

Motivations

- Distributed span representations are useful in various NLP tasks
- Existing methods are mostly based on simple derivations from word or sub-word representations.
- We improve span representations using the underlying compositional structures of text, which are often represented with constituency parse trees.

Existing Methods

- **Simple derivations:**
  - From word or sub-word representations, such as average, max-pooling. These methods ignore the compositional structures both within and outside text spans.
- **Recursive neural networks (RvNNs):**
  - Recursively compose span representations from their sub-spans but separate the representations computed from both directions and disallow them to directly interact with each other.
- **Graph neural networks (GNNs):**
  - Such as GCN and GAT, represent each composition with multiple edges that become mixed up with edges from other compositions.

Our Method: Hypertree Neural Networks (HTNN)

- **View A Constituency Parse Tree as A Hypertree**

Each node represents a constituent and each hyperedge is a tuple of multiple nodes representing a composition of smaller child constituents into a larger parent constituent .

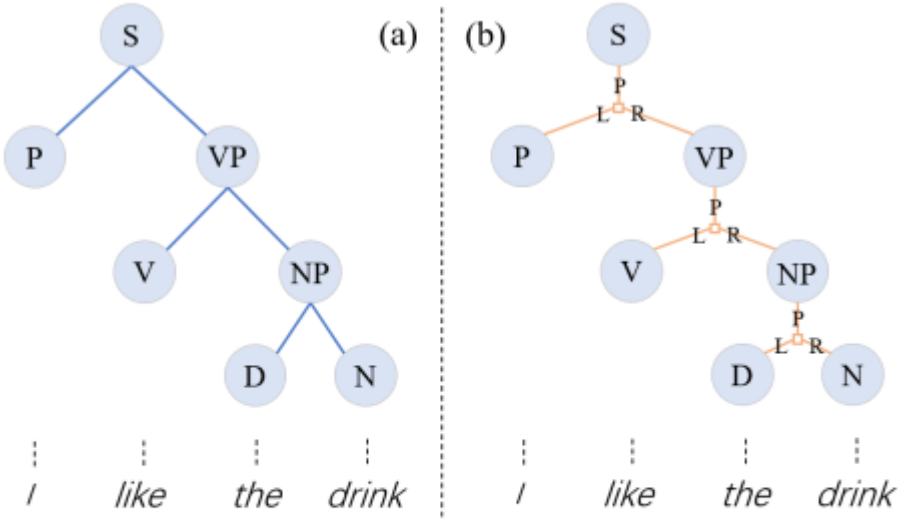

Figure 1: An example binarized constituency parse tree (a) and its corresponding hypertree (b). For each hyperedge, P/L/R denote the parent, left-child, right-child constituent spans respectively.

Our Method: Hypertree Neural Networks (HTNN)

- **Initialization of Node Representation**
  - For a span s = [i, j], we use attention pooling method to initialize the representations

$$\mathbf{s}_{ij} = \sum_{k=i}^{j-1} a_k \cdot \mathbf{e}_k \qquad a_k = \mathbf{Softmax}(\mathbf{v}_1^T \cdot \mathbf{e}_k)$$

  - Concatenate with constituent tag embedding

$$\mathbf{s}'_{ij} = \mathbf{Concat}([\mathbf{s}_{ij}; \mathbf{Embedding}(tag)])$$

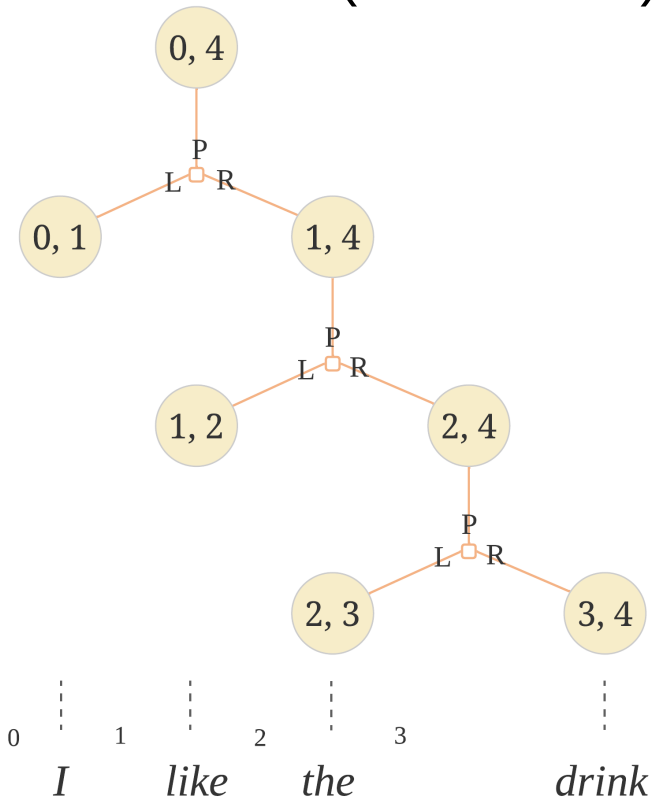

Figure 2: Initialize nodes' representations with word embeddings

Our Method: Hypertree Neural Networks (HTNN)

- **Composition within Hyperedge**
  - Any node can be computed from the others within each hyperedge.

$$[\mathbf{h}'_p; \mathbf{c}'_p] = \mathbf{Compose}(\mathbf{h}_p, \mathbf{h}_l, \mathbf{h}_r, \mathbf{c}_l, \mathbf{c}_r)$$
$$[\mathbf{h}'_l; \mathbf{c}'_l] = \mathbf{Compose}(\mathbf{h}_l, \mathbf{h}_p, \mathbf{h}_r, \mathbf{c}_p, \mathbf{c}_r)$$
$$[\mathbf{h}'_r; \mathbf{c}'_r] = \mathbf{Compose}(\mathbf{h}_r, \mathbf{h}_p, \mathbf{h}_l, \mathbf{c}_p, \mathbf{c}_l)$$

  - The composition function is inspired by TreeLSTM

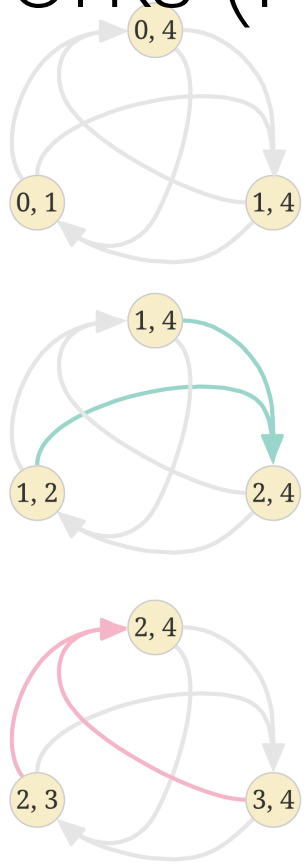

Figure 3: Composition process within three hyperedges. The node (2,4) can be computed in two hyperedges

Our Method: Hypertree Neural Networks (HTNN)

- **Aggregation of Multiple Representations**
  - A node in the HTNN is connected with two hyperedges, resulting in two different representations for the node.
  - A third representation is from the previous layer

$$a_i = \mathbf{Softmax}(\mathbf{v}_2^T \mathbf{tanh}(\mathbf{W}[\mathbf{h}'_0; \mathbf{h}'_i]))$$

$$\mathbf{h}' = \sum_i a_i \cdot \mathbf{h}'_i, \quad i \in \{0, 1, 2\}$$

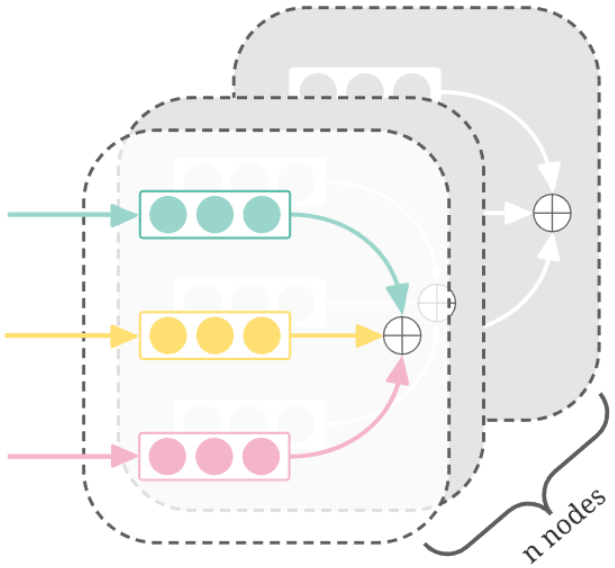

Figure 4: Aggregation representations from three different sources

Our Method: Hypertree Neural Networks (HTNN)

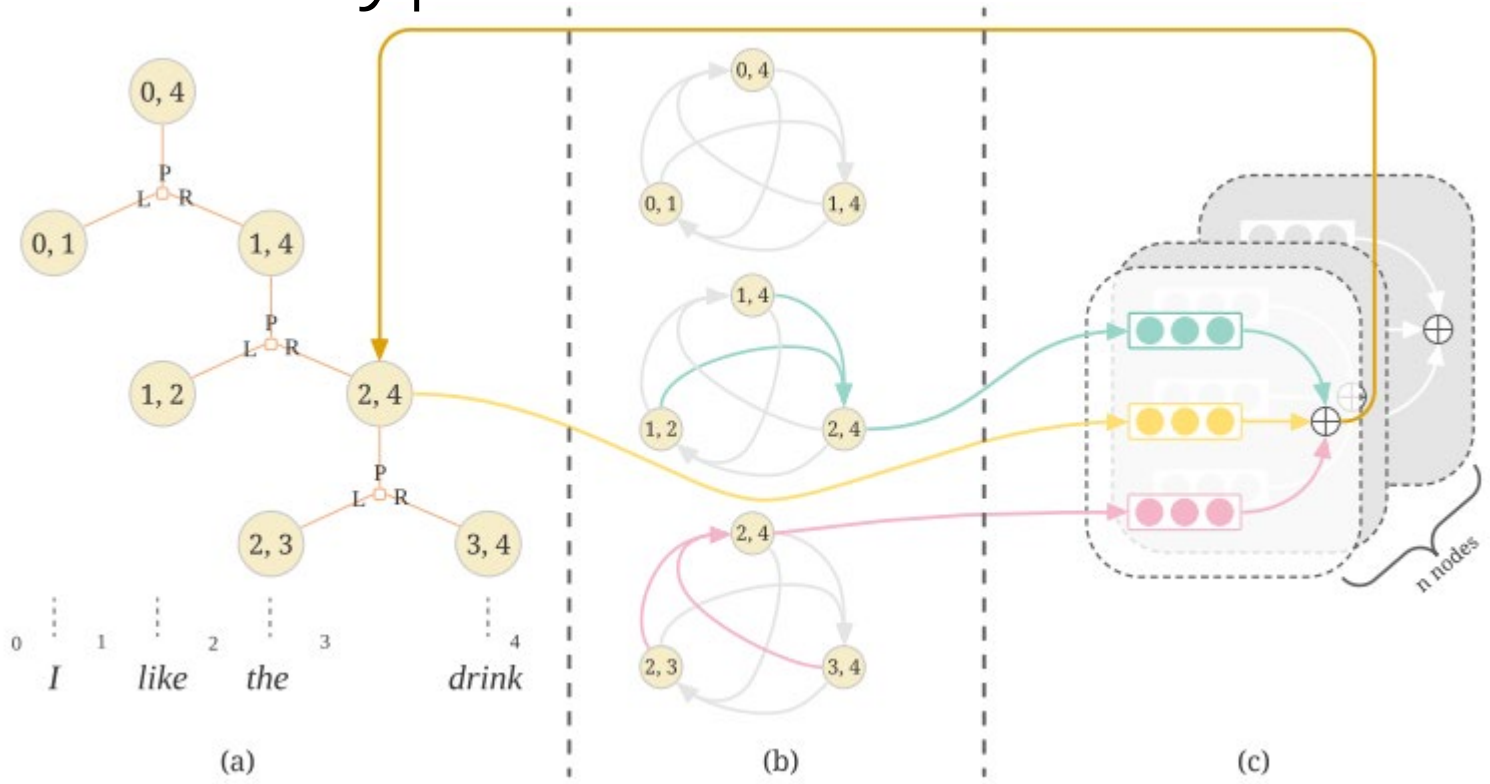

Figure 5: Illustration of HTNN, a iteratively updating method.

Experiments & Results

- **Probing Experiments**
  - HTNN shows strong performance on Named entity labeling (NEL), Semantic role classification (SRC), Coreference arc prediction (COREF)

	NEL		SRC		COREF		AVG	
	F1-const	F1-all	F1-const	F1-all	F1-const	F1-all	F1-const	F1-all
Pooling	96.18	**96.07**	93.08	93.05	92.99	93.01	94.08	94.04
TreeLSTM	95.05	94.22	90.02	89.88	90.07	89.70	91.71	91.27
Bi-TreeLSTM	95.25	94.42	90.49	90.35	90.33	89.96	92.02	91.58
SentiBERT	92.98	92.84	95.92	95.09	96.01	95.64	94.97	94.52
GAT	96.01	95.18	93.41	93.26	96.13	95.76	95.18	94.73
GCN	96.03	95.20	93.51	93.37	96.22	95.85	95.25	94.81
GAT-sib	95.79	94.96	92.85	92.71	95.66	95.29	94.77	94.32
GCN-sib	95.87	95.04	93.27	93.13	95.68	95.31	94.94	94.50
HTNN	**96.28**	95.45	**93.88**	**93.74**	**96.33**	**95.96**	**95.50**	**95.05**

Table 1: Results of probing experiments.

Experiments & Results

- **Semantic Role Labeling (SRL) Experiments**
  - HTNN achieves the best performance on all the three datasets.

	CONLL12		CONLL05 WSJ		CONLL05 BROWN	
	F1-const	F1-all	F1-const	F1-all	F1-const	F1-all
Pooling	82.36	82.23	82.82	81.90	71.51	70.43
SentiBERT	75.31	75.18	74.52	73.69	66.52	65.51
GAT	85.29	85.16	84.69	83.74	73.32	72.24
GCN	87.91	87.77	88.06	87.08	79.22	78.09
GAT-sib	76.41	76.29	78.34	77.46	62.52	61.58
GCN-sib	88.40	88.27	88.45	87.46	80.03	79.87
HTNN	**89.94**	**89.81**	**90.77**	**89.76**	**82.88**	**81.68**
Wang et al. (2019)†	-	84.21	-	85.23	-	75.36
Fei et al. (2021)*	-	87.35	-	88.81	-	81.27

Table 2: Results of SRL experiments. "†": reimplemented and reported by Fei et al. (2021). "∗": results reported in the original paper.

Conclusions & Future Work

- We propose hypertree neural networks (HTNN) to generate better representations of constituent spans following constituency parse tree structures.

- In the future, we plan to tackle two related issues:
  - Reliance on high-quality constituency parses
  - Inability to represent distituent spans