Neural Tree Indexers for Text Understanding

Tsendsuren Munkhdalai and **Hong Yu**
University of Massachusetts, MA, USA
{tsendsuren.munkhdalai,hong.yu}@umassmed.edu

Abstract

Recurrent neural networks (RNNs) process input text sequentially and model the conditional transition between word tokens. In contrast, the advantages of recursive networks include that they explicitly model the compositionality and the recursive structure of natural language. However, the current recursive architecture is limited by its dependence on syntactic tree. In this paper, we introduce a robust syntactic parsing-independent tree structured model, Neural Tree Indexers (NTI) that provides a middle ground between the sequential RNNs and the syntactic tree-based recursive models. NTI constructs a *full n-ary tree* by processing the input text with its node function in a bottom-up fashion. Attention mechanism can then be applied to both structure and node function. We implemented and evaluated a binary-tree model of NTI, showing the model achieved the state-of-the-art performance on three different NLP tasks: natural language inference, answer sentence selection, and sentence classification, outperforming state-of-the-art recurrent and recursive neural networks [1].

1 Introduction

Recurrent neural networks (RNNs) have been successful for modeling sequence data (Elman, 1990). RNNs equipped with gated hidden units and internal short-term memories, such as long short-term memories (LSTM) (Hochreiter and Schmidhuber, 1997) have achieved a notable success in several NLP tasks including named entity recognition (Lample et al., 2016), constituency parsing (Vinyals et al., 2015), textual entailment recognition (Rocktäschel et al., 2016), question answering (Hermann et al., 2015), and machine translation (Bahdanau et al., 2015). However, most LSTM models explored so far are sequential. It encodes text sequentially from left to right or vice versa and do not naturally support compositionality of language. Sequential LSTM models seem to learn syntactic structure from the natural language however their generalization on unseen text is relatively poor comparing with models that exploit syntactic tree structure (Bowman et al., 2015b).

Unlike sequential models, recursive neural networks compose word phrases over syntactic tree structure and have shown improved performance in sentiment analysis (Socher et al., 2013). However its dependence on a syntactic tree architecture limits practical NLP applications. In this study, we introduce Neural Tree Indexers (NTI), a class of tree structured models for NLP tasks. NTI takes a sequence of tokens and produces its representation by constructing a *full n-ary tree* in a bottom-up fashion. Each node in NTI is associated with one of the node transformation functions: leaf node mapping and non-leaf node composition functions. Unlike previous recursive models, the tree structure for NTI is relaxed, i.e., NTI does not require the input sequences to be parsed syntactically; and therefore it is flexible and can be directly applied to a wide range of NLP tasks beyond sentence modeling.

Furthermore, we propose different variants of node composition function and attention over tree for our NTI models. When a sequential leaf node transformer such as LSTM is chosen, the NTI network forms a sequence-tree hybrid model taking advantage of both conditional and compositional powers of sequential and recursive models. Figure

[1] Code for the experiments and NTI is available at https://bitbucket.org/tsendeemts/nti

Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers, pages 11–21,
Valencia, Spain, April 3-7, 2017.

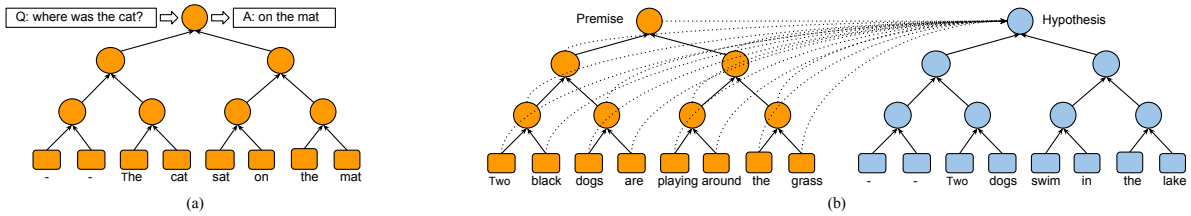

Figure 1: A binary tree form of Neural Tree Indexers (NTI) in the context of question answering and natural language inference. We insert empty tokens (denoted by $-$) to the input text to form a full binary tree. (a) NTI produces answer representation at the root node. This representation along with the question is used to find the answer. (b) NTI learns representations for the premise and hypothesis sentences and then attentively combines them for classification. Dotted lines indicate attention over premise-indexed tree.

1 shows a binary-tree model of NTI. Although the model does not follow the syntactic tree structure, we empirically show that it achieved the state-of-the-art performance on three different NLP applications: natural language inference, answer sentence selection, and sentence classification.

2 Related Work

2.1 Recurrent Neural Networks and Attention Mechanism

RNNs model input text sequentially by taking a single token at each time step and producing a corresponding hidden state. The hidden state is then passed along through the next time step to provide historical sequence information. Although a great success in a variety of tasks, RNNs have limitations (Bengio et al., 1994; Hochreiter, 1998). Among them, it is not efficient at memorizing long or distant sequence (Sutskever et al., 2014). This is frequently called as information flow bottleneck. Approaches have therefore been developed to overcome the limitations. For example, to mitigate the information flow bottleneck, Bahdanau et al. (2015) extended RNNs with a soft attention mechanism in the context of neural machine translation, leading to improved the results in translating longer sentences.

RNNs are linear chain-structured; this limits its potential for natural language which can be represented by complex structures including syntactic structure. In this study, we propose models to mitigate this limitation.

2.2 Recursive Neural Networks

Unlike RNNs, recursive neural networks explicitly model the compositionality and the recursive structure of natural language over tree. The tree structure can be predefined by a syntactic parser (Socher et al., 2013). Each non-leaf tree node is associated with a node composition function which combines its children nodes and produces its own representation. The model is then trained by back-propagating error through structures (Goller and Kuchler, 1996).

The node composition function can be varied. A single layer network with $tanh$ non-linearity was adopted in recursive auto-associate memories (Pollack, 1990) and recursive autoencoders (Socher et al., 2011). Socher et al. (2012) extended this network with an additional matrix representation for each node to augment the expressive power of the model. Tensor networks have also been used as composition function for sentence-level sentiment analysis task (Socher et al., 2013). Recently, Zhu et al. (2015) introduced S-LSTM which extends LSTM units to compose tree nodes in a recursive fashion.

In this paper, we introduce a novel attentive node composition function that is based on S-LSTM. Our NTI model does not rely on either a parser output or a fine-grained supervision of non-leaf nodes, both required in previous work. In NTI, the supervision from the target labels is provided at the root node. As such, our NTI model is robust and applicable to a wide range of NLP tasks. We introduce attention over tree in NTI to overcome the vanishing/explode gradients challenges as shown in RNNs.

3 Methods

Our training set consists of N examples $\{X^i, Y^i\}_{i=1}^N$, where the input X^i is a sequence of word tokens $w_1^i, w_2^i, \ldots, w_{T_i}^i$ and the output Y^i can be either a single target or a sequence. Each

input word token w_t is represented by its word embedding $x_t \in R^k$.

NTI is a *full n-ary tree* (and the sub-trees can be overlapped). It has two types of transformation function: non-leaf node function $f^{node}(h^1, \ldots, h^c)$ and leaf node function $f^{leaf}(x_t)$. $f^{leaf}(x_t)$ computes a (possibly non-linear) transformation of the input word embedding x_t. $f^{node}(h^1, \ldots, h^c)$ is a function of its child nodes representation $h^1, \ldots, h^c$, where c is the total number of child nodes of this non-leaf node.

NTI can be implemented with different tree structures. In this study we implemented and evaluated a binary tree form of NTI: a non-leaf node can take in only two direct child nodes (i.e., $c = 2$). Therefore, the function $f^{node}(h^l, h^r)$ composes its left child node h^l and right child node h^r. Figure 1 illustrates our NTI model that is applied to question answering (a) and natural language inference tasks (b). Note that the node and leaf node functions are neural networks and are the only training parameters in NTI.

We explored two different approaches to compose node representations: an extended LSTM and attentive node composition functions, to be described below.

3.1 Non-Leaf Node Composition Functions

We define two different methods for non-leaf node function $f^{node}(h^l, h^r)$.

LSTM-based Non-leaf Node Function (S-LSTM): We initiate $f^{node}(h^l, h^r)$ with LSTM. For non-leaf node, we adopt S-LSTM Zhu et al. (2015), an extension of LSTM to tree structures, to learn a node representation by its children nodes. Let h_t^l, h_t^r, c_t^l and c_t^r be vector representations and cell states for the left and right children. An S-LSTM computes a parent node representation h_{t+1}^p and a node cell state c_{t+1}^p as

$$i_{t+1} = \sigma(W_1^s h_t^l + W_2^s h_t^r + W_3^s c_t^l + W_4^s c_t^r) \quad (1)$$

$$f_{t+1}^l = \sigma(W_5^s h_t^l + W_6^s h_t^r + W_7^s c_t^l + W_8^s c_t^r) \quad (2)$$

$$f_{t+1}^r = \sigma(W_9^s h_t^l + W_{10}^s h_t^r + W_{11}^s c_t^l + W_{12}^s c_t^r) \quad (3)$$

$$c_{t+1}^p = f_{t+1}^l \odot c_t^l + f_{t+1}^r \odot c_t^r + i_{t+1} \odot \tanh(W_{13}^s h_t^l + W_{14}^s h_t^r) \quad (4)$$

$$o_{t+1} = \sigma(W_{15}^s h_t^l + W_{16}^s h_t^r + W_{18}^s c_{t+1}^p) \quad (5)$$

$$h_{t+1}^p = o_{t+1} \odot tanh(c_{t+1}^p) \quad (6)$$

where $W_1^s, \ldots, W_{18}^s \in R^{k \times k}$ and biases (for brevity we eliminated the bias terms) are the training parameters. σ and $\odot$ denote the element-wise $sigmoid$ function and the element-wise vector multiplication. Extension of S-LSTM non-leaf node function to compose more children is straightforward. However, the number of parameters increases quadratically in S-LSTM as we add more child nodes.

Attentive Non-leaf Node Function (ANF): Some NLP applications (e.g., QA and machine translation) would benefit from a dynamic query dependent composition function. We introduce ANF as a new non-leaf node function. Unlike S-LSTM, ANF composes the child nodes attentively in respect to another relevant input vector $q \in R^k$. The input vector q can be a learnable representation from a sequence representation. Given a matrix $S^{ANF} \in R^{k \times 2}$ resulted by concatenating the child node representations h_t^l, h_t^r and the third input vector q, ANF is defined as

$$m = f^{score}(S^{ANF}, q) \quad (7)$$

$$\alpha = softmax(m) \quad (8)$$

$$z = S^{ANF} \alpha^\top \quad (9)$$

$$h_{t+1}^p = ReLU(W_1^{ANF} z) \quad (10)$$

where $W_1^{ANF} \in R^{k \times k}$ is a learnable matrix, $m \in R^2$ the attention score and $\alpha \in R^2$ the attention weight vector for each child. f^{score} is an attention scoring function, which can be implemented as a multi-layer perceptron (MLP)

$$m = w^\top ReLU(W_1^{score} S^{ANF} + W_2^{score} q \otimes e) \quad (11)$$

or a matrix-vector product $m = q^\top S^{ANF}$. The matrices W_1^{score} and $W_2^{score} \in R^{k \times k}$ and the vector $w \in R^k$ are training parameters. $e \in R^2$ is a vector of ones and $\otimes$ the outer product. We use $ReLU$ function for non-linear transformation.

3.2 Attention Over Tree

Comparing with sequential LSTM models, NTI has less recurrence, which is defined by the tree depth, $log(n)$ for binary tree where n is the length of the input sequence. However, NTI still needs to compress all the input information into a single representation vector of the root. This imposes practical difficulties when processing long sequences. We address this issue with attention

Model	d	$\|\theta\|_M$	Train	Test
Classifier with handcrafted features (Bowman et al., 2015a)	-	-	99.7	78.2
LSTMs encoders (Bowman et al., 2015a)	300	3.0M	83.9	80.6
Dependency Tree CNN encoders (Mou et al., 2016)	300	3.5M	83.3	82.1
NTI-SLSTM (Ours)	300	3.3M	83.9	82.4
SPINN-PI encoders (Bowman et al., 2016)	300	3.7M	89.2	83.2
NTI-SLSTM-LSTM (Ours)	300	4.0M	82.5	83.4
LSTMs attention (Rocktäschel et al., 2016)	100	242K	85.4	82.3
LSTMs word-by-word attention (Rocktäschel et al., 2016)	100	250K	85.3	83.5
NTI-SLSTM node-by-node global attention (Ours)	300	3.5M	85.0	84.2
NTI-SLSTM node-by-node tree attention (Ours)	300	3.5M	86.0	84.3
NTI-SLSTM-LSTM node-by-node tree attention (Ours)	300	4.2M	88.1	85.7
NTI-SLSTM-LSTM node-by-node global attention (Ours)	300	4.2M	87.6	85.9
mLSTM word-by-word attention (Wang and Jiang, 2016)	300	1.9M	92.0	86.1
LSTMN with deep attention fusion (Cheng et al., 2016)	450	3.4M	88.5	86.3
Tree matching NTI-SLSTM-LSTM tree attention (Ours)	300	3.2M	87.3	86.4
Decomposable Attention Model (Parikh et al., 2016)	200	580K	90.5	86.8
Tree matching NTI-SLSTM-LSTM global attention (Ours)	300	3.2M	87.6	87.1
Full tree matching NTI-SLSTM-LSTM global attention (Ours)	300	3.2M	88.5	**87.3**

Table 1: Training and test accuracy on natural language inference task. d is the word embedding size and $|\theta|_M$ the number of model parameters.

mechanism over tree. In addition, the attention mechanism can be used for matching trees (described in Section 4 as Tree matching NTI) that carry different sequence information. We first define a global attention and then introduce a tree attention which considers the parent-child dependency for calculation of the attention weights.

Global Attention: An attention neural network for the global attention takes all node representations as input and produces an attentively blended vector for the whole tree. This neural net is similar to ANF. Particularly, given a matrix $S^{GA} \in R^{k \times 2n-1}$ resulted by concatenating the node representations $h_1, \ldots, h_{2n-1}$ and the relevant input representation q, the global attention is defined as

$$m = f^{score}(S^{GA}, q) \tag{12}$$

$$\alpha = softmax(m) \tag{13}$$

$$z = S^{GA}\alpha^\top \tag{14}$$

$$h^{tree} = ReLU(W_1^{GA} z + W_2^{GA} q) \tag{15}$$

where W_1^{GA} and $W_2^{GA} \in R^{k \times k}$ are training parameters and $\alpha \in R^{2n-1}$ the attention weight vector for each node. This attention mechanism is robust as it globally normalizes the attention score m with $softmax$ to obtain the weights α. However, it does not consider the tree structure when producing the final representation h^{tree}.

Tree Attention: We modify the global attention network to the tree attention mechanism. The resulting tree attention network performs almost the same computation as ANF for each node. It compares the parent and children nodes to produce a new representation assuming that all node representations are constructed. Given a matrix $S^{TA} \in R^{k \times 3}$ resulted by concatenating the parent node representation h_t^p, the left child h_t^l and the right child h_t^r and the relevant input representation q, every non-leaf node h_t^p simply updates its own representation by using the following equation in a bottom-up manner.

$$m = f^{score}(S^{TA}, q) \tag{16}$$

$$\alpha = softmax(m) \tag{17}$$

$$z = S^{TA}\alpha^\top \tag{18}$$

$$h_t^p = ReLU(W_1^{TA} z) \tag{19}$$

and this equation is similarity to the global attention. However, now each non-leaf node attentively collects its own and children representations and passes towards the root which finally constructs the attentively blended tree representation. Note that unlike the global attention, the tree attention locally normalizes the attention scores with $softmax$.

4 Experiments

We describe in this section experiments on three different NLP tasks, natural language inference, question answering and sentence classification to demonstrate the flexibility and the effectiveness of NTI in the different settings.

We trained NTI using Adam (Kingma and Ba, 2014) with hyperparameters selected on development set. The pre-trained 300-D Glove 840B vectors (Pennington et al., 2014) were obtained for the word embeddings[2]. The word embeddings are fixed during training. The embeddings for out-of-vocabulary words were set to zero vector. We pad the input sequence to form a *full binary tree*. A padding vector was inserted when padding. We analyzed the effects of the padding size and found out that it has no influence on the performance (see Appendix 5.3). The size of hidden units of the NTI modules were set to 300. The models were regularized by using dropouts and an l_2 weight decay.[3]

4.1 Natural Language Inference

We conducted experiments on the Stanford Natural Language Inference (SNLI) dataset (Bowman et al., 2015a), which consists of 549,367/9,842/9,824 premise-hypothesis pairs for train/dev/test sets and target label indicating their relation. Unless otherwise noted, we follow the setting in the previous work (Mou et al., 2016; Bowman et al., 2016) and use an MLP for classification which takes in NTI outputs and computes the concatenation $[h^p_{2n-1}; h^h_{2n-1}]$, absolute difference $h^p_{2n-1} - h^h_{2n-1}$ and elementwise product $h^p_{2n-1} \cdot h^h_{2n-1}$ of the two sentence representations. The MLP has also an input layer with 1024 units with $ReLU$ activation and a $softmax$ output layer. We explored nine different task-oriented NTI models with varying complexity, to be described below. For each model, we set the batch size to 32. The initial learning, the regularization strength and the number of epoch to be trained are varied for each model.

NTI-SLSTM: this model does not rely on f^{leaf} transformer but uses the S-LSTM units for the non-leaf node function. We set the initial learning rate to 1e-3 and l_2 regularizer strength to 3e-5, and train the model for 90 epochs. The neural net was regularized by 10% input dropouts and the 20% output dropouts.

NTI-SLSTM-LSTM: we use LSTM for the leaf node function f^{leaf}. Concretely, the LSTM output vectors are given to NTI-SLSTM and the memory cells of the lowest level S-LSTM were initialized with the LSTM memory states. The hyper-parameters are the same as the previous model.

NTI-SLSTM node-by-node global attention: This model learns inter-sentence relation with the global attention over premise-indexed tree, which is similar to word-by-word attention model of Rocktäschel et al. (2016) in that it attends over the premise tree nodes at every time step of hypothesis encoding. We tie the weight parameters of the two NTI-SLSTMs for premise and hypothesis and no f^{leaf} transformer used. We set the initial learning rate to 3e-4 and l_2 regularizer strength to 1e-5, and train the model for 40 epochs. The neural net was regularized by 15% input dropouts and the 15% output dropouts.

NTI-SLSTM node-by-node tree attention: this is a variation of the previous model with the tree attention. The hyper-parameters are the same as the previous model.

NTI-SLSTM-LSTM node-by-node global attention: in this model we include LSTM as the leaf node function f^{leaf}. Here we initialize the memory cell of S-LSTM with LSTM memory and hidden/memory state of hypothesis LSTM with premise LSTM (the later follows the work of (Rocktäschel et al., 2016)). We set the initial learning rate to 3e-4 and l_2 regularizer strength to 1e-5, and train the model for 10 epochs. The neural net was regularized by 10% input dropouts and the 15% output dropouts.

NTI-SLSTM-LSTM node-by-node tree attention: this is a variation of the previous model with the tree attention. The hyper-parameters are the same as the previous model.

Tree matching NTI-SLSTM-LSTM global attention: this model first constructs the premise and hypothesis trees simultaneously with the NTI-SLSTM-LSTM model and then computes their matching vector by using the global attention and an additional LSTM. The attention vectors are produced at each hypothesis tree node and then are given to the LSTM model sequentially. The LSTM model compress the attention vectors and outputs a single matching vector, which is passed to an MLP for classification. The MLP for this tree matching setting has an input layer with 1024 units with $ReLU$ activation and a $softmax$ output layer.

Unlike Wang and Jiang (2016)'s matching LSTM model which is specific to matching sequences, we use the standard LSTM units and match trees. We set the initial learning rate to 3e-

[2] http://nlp.stanford.edu/projects/glove/

[3] More detail on hyper-parameters can be found in code.

4 and l_2 regularizer strength to 3e-5, and train the model for 20 epochs. The neural net was regularized by 20% input dropouts and the 20% output dropouts.

Tree matching NTI-SLSTM-LSTM tree attention: we replace the global attention with the tree attention. The hyper-parameters are the same as the previous model.

Full tree matching NTI-SLSTM-LSTM global attention: this model produces two sets of the attention vectors, one by attending over the premise tree regarding each hypothesis tree node and another by attending over the hypothesis tree regarding each premise tree node. Each set of the attention vectors is given to a LSTM model to achieve full tree matching. The last hidden states of the two LSTM models (i.e. one for each attention vector set) are concatenated for classification. The training weights are shared among the LSTM models The hyper-parameters are the same as the previous model.[4]

Table 1 shows the results of our models. For comparison, we include the results from the published state-of-the-art systems. While most of the sentence encoder models rely solely on word embeddings, the dependency tree CNN and the SPINN-PI models make use of sentence parser output; which present strong baseline systems. The last set of methods designs inter-sentence relation with soft attention (Bahdanau et al., 2015). Our best score on this task is 87.3% accuracy obtained with the full tree matching NTI model. The previous best performing model on the task performs phrase matching by using the attention mechanism.

Our results show that NTI-SLSTM improved the performance of the sequential LSTM encoder by approximately 2%. Not surprisingly, using LSTM as leaf node function helps in learning better representations. Our NTI-SLSTM-LSTM is a hybrid model which encodes a sequence sequentially through its leaf node function and then hierarchically composes the output representations. The node-by-node attention models improve the performance, indicating that modeling inter-sentence interaction is an important element in NLI. Aggregating matching vector between trees or sequences with a separate LSTM model is effective. The global attention seems to be robust on this task. The tree attention were not helpful as it normalizes the attention scores locally in parent-child relationship.

[4]Computational constraint prevented us from experimenting the tree attention variant of this model

Model	MAP	MRR
Classifier with features (2013)	0.5993	0.6068
Paragraph Vector (2014)	0.5110	0.5160
Bigram-CNN (2014)	0.6190	0.6281
3-layer LSTM (2016)	0.6552	0.6747
3-layer LSTM attention (2016)	0.6639	0.6828
NASM (2016)	0.6705	**0.6914**
NTI (Ours)	**0.6742**	0.6884

Table 2: Test set performance on answer sentence selection.

Model	Bin	FG
RNTN (Socher et al., 2013)	85.4	45.7
CNN-MC (Kim, 2014)	88.1	47.4
DRNN (Irsoy and Cardie, 2015)	86.6	49.8
2-layer LSTM (Tai et al., 2015)	86.3	46.0
Bi-LSTM (Tai et al., 2015)	87.5	49.1
NTI-SLSTM (Ours)	87.8	50.5
CT-LSTM (Tai et al., 2015)	88.0	51.0
DMN (Kumar et al., 2016)	88.6	52.1
NTI-SLSTM-LSTM (Ours)	**89.3**	**53.1**

Table 3: Test accuracy for sentence classification. Bin: binary, FG: fine-grained 5 classes.

4.2 Answer Sentence Selection

For this task, a model is trained to identify the correct sentences that answer a factual question, from a set of candidate sentences. We experiment on WikiQA dataset constructed from Wikipedia (Yang et al., 2015). The dataset contains 20,360/2,733/6,165 QA pairs for train/dev/test sets.

We used the same setup in the language inference task except that we replace the $softmax$ layer with a $sigmoid$ layer and model the following conditional probability distribution.

$$p_\theta(y = 1|h_n^q, h_n^a) = sigmoid(o^{QA}) \qquad (20)$$

where h_n^q and h_n^a are the question and the answer encoded vectors and o^{QA} denotes the output of the hidden layer of the MLP. For this task, we use NTI-SLSTM-LSTM to encode answer candidate sentences and NTI-ANF-LSTM to encode the question sentences. Note that NTI-ANF-LSTM is relied on ANF as the non-leaf node function. q vector for NTI-ANF-LSTM is the answer representation produced by the answer encoding NTI-SLSTM-LSTM model. We set the batch size to 4 and the initial learning rate to 1e-3, and train the

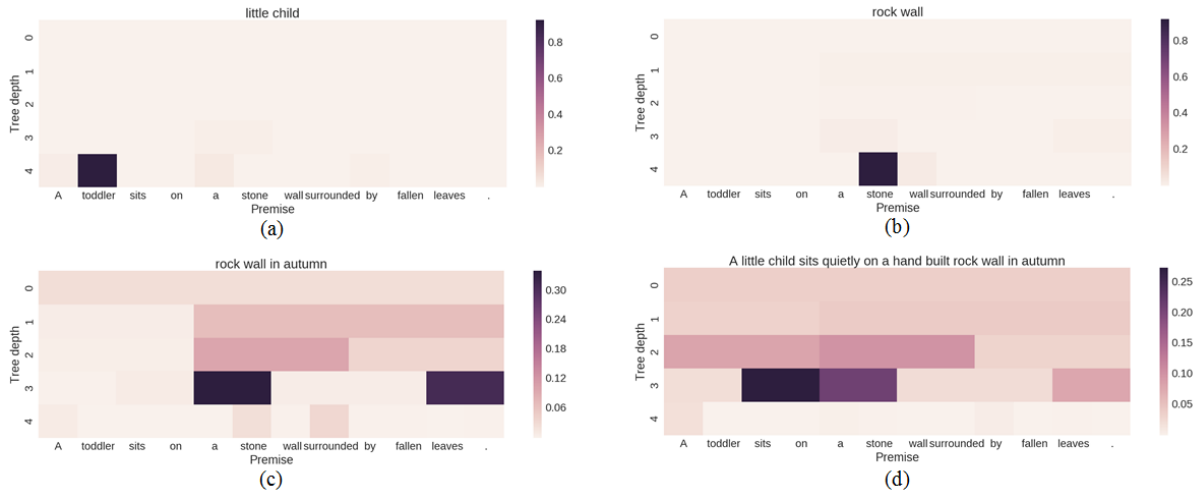

Figure 2: Node-by-node attention visualizations. The phrases shown on the top are nodes from hypothesis-indexed tree and the premise tokens are listed along the x-axis. The adjacent cells are composed in the top cell representing a binary tree and resulting a longer attention span.

a person	**park for fun**	**Santa Claus**	**sad, depressed, and hatred**
single person	an outdoor concert at the park	a snowmobile in a blizzard	an Obama supporter is upset
a woman	kids playing at a park outside	a Skier ski - jumping	but doesn't have any money
a young person	a mom takes a break in a park	A skier preparing a trick	crying because he didn't get cake
a guy	people play frisbee outdoors	a child is playing on christmas	trying his hardest to not fall off
a single human	takes his lunch break in the park	two men play with a snowman	is upset and crying on the ground

Table 4: Nearest-neighbor phrases based on cosine similarity between learned representations.

model for 10 epochs. We used 20% input dropouts and no l_2 weight decay. Following previous work, we adopt MAP and MRR as the evaluation metrics for this task.[5]

Table 2 presents the results of our model and the previous models for the task.[6] The classifier with handcrafted features is a SVM model trained with a set of features. The Bigram-CNN model is a simple convolutional neural net. The Deep LSTM and LSTM attention models outperform the previous best result by a large margin, nearly 5-6%. NASM improves the result further and sets a strong baseline by combining variational auto-encoder (Kingma and Welling, 2014) with the soft attention. In NASM, they adopt a deep three-layer LSTM and introduced a latent stochastic attention mechanism over the answer sentence. Our NTI model exceeds NASM by approximately 0.4% on MAP for this task.

[5]We used *trec_eval* script to calculate the evaluation metrics

[6]Inclusion of simple word count feature improves the performance by around 0.15-0.3 across the board

4.3 Sentence Classification

Lastly, we evaluated NTI on the Stanford Sentiment Treebank (SST) (Socher et al., 2013). This dataset comes with standard train/dev/test sets and two subtasks: binary sentence classification or fine-grained classification of five classes. We trained our model on the text spans corresponding to labeled phrases in the training set and evaluated the model on the full sentences.

We use NTI-SLSTM and NTI-SLSTM-LSTM models to learn sentence representations for the task. The sentence representations were passed to a two-layer MLP for classification. We set the batch size to 64, the initial learning rate to 1e-3 and l_2 regularizer strength to 3e-5, and train each model for 10 epochs. The NTI-SLSTM model was regularized by 10%/20% of input/output and 20%/30% of input/output dropouts and the NTI-SLSTM-LSTM model 20% of input and 20%/30% of input/output dropouts for binary and fine-grained settings.

NTI-SLSTM-LSTM (as shown in Table 5) set the state-of-the-art results on both subtasks. Our NTI-SLSTM model performed slightly worse

A dog mouth holds a retrieved ball.	**A cat nurses puppies.**	**A dog sells a woman a hat.**
A brown and white dog holds a tennis ball in his mouth.	A golden retriever nurses some other dogs puppies.	The dog is a labrador retriever.
The dog has a ball.	A golden retriever nurses puppies.	A girl is petting her dog.
The dogs are chasing a ball.	A mother dog checking up on her baby puppy.	The dog is a shitzu.
A small dog runs to catch a ball.	A girl is petting her dog.	A husband and wife making pizza.
The puppy is chasing a ball.	The hat wearing girl is petting a cat.	The dog is a chihuahua.

Table 5: Nearest-neighbor sentences based on cosine similarity between learned representations.

than its constituency tree-based counter part, CT-LSTM model. The CT-LSTM model composes phrases according to the output of a sentence parser and uses a node composition function similar to S-LSTM. After we transformed the input with the LSTM leaf node function, we achieved the best performance on this task.

5 Qualitative Analysis

5.1 Attention and Compositionality

To help analyzing the results, we output attention weights by our NTI-SLSTM node-by-node global attention model. Figure 2 shows the attention heatmaps for two sentences in the SNLI test set. It shows that our model semantically aligns single or multiword expressions (*"little child"* and *"toddler"*; *"rock wall"* and *"stone"*). In addition, our model is able to re-orient its attention over different parts of the hypothesis when the expression is more complex. For example, for (c) *"rock wall in autumn"*, NTI mostly focuses on the nodes in depth 1, 2 and 3 representing contexts related to *"a stone"*, *"leaves."* and *"a stone wall surrounded"*. Surprisingly, attention degree for the single word expression like *"stone"*, *"wall"* and *"leaves"* is lower to compare with multiword phrases. Sequence models lack this property as they have no explicit composition module to produce such multiword phrases.

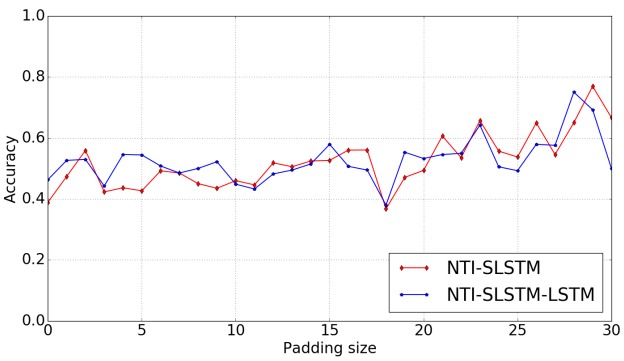

Figure 3: Fine-grained sentiment classification accuracy vs. padding size on test set of SST data.

Finally, the most interesting pattern is that the model attends over higher level (low depth) tree nodes with rich semantics when considering a (c) longer phrase or (d) full sentence. As shown in (d), the NTI model aligns the root node representing the whole hypothesis sentence to the higher level tree nodes covering larger sub-trees in the premise. It certainly ignores the lower level single word expressions and only starts to attend when the words are collectively to form rich semantics.

5.2 Learned Representations of Phrases and Sentences

Using cosine similarity between their representations produced by the NTI-SLSTM model, we show that NTI is able to capture paraphrases on SNLI test data. As shown in Table 4, NTI seems to distinguish plural from singular forms (similar phrases to *"a person"*). In addition, NTI captures non-surface knowledge. For example, the phrases similar to *"park for fun"* tend to align to the semantic content of *fun* and *park*, including *"people play frisbee outdoors"*. The NTI model was able to relate *"Santa Claus"* to *christmas* and *snow*. Interestingly, the learned representations were also able to connect implicit semantics. For example, NTI found that *"sad, depressed, and hatred"* is close to the phrases like *"an Obama supporter is upset"*. Overall the NTI model is robust to the length of the phrases being matched. Given a short phrase, NTI can retrieve longer yet semantically coherent sequences from the SNLI test set.

In Table 5, we show nearest-neighbor sentences from SNLI test set. Note that the sentences listed in the first two columns sound semantically coherent but not the ones in the last column. The query sentence *"A dog sells a women a hat"* does not actually represent a common-sense knowledge and this sentence now seem to confuse the NTI model. As a result, the retrieved sentence are arbitrary and not coherent.

5.3 Effects of Padding Size

We introduced a special padding character in order to construct full binary tree. Does this padding character influence the performance of the NTI models? In Figure 3, we show relationship between the padding size and the accuracy on Stanford sentiment analysis data. Each sentence was padded to form a full binary tree. The x-axis represents the number of padding characters introduced. When the padding size is less (up to 10), the NTI-SLSTM-LSTM model performs better. However, this model tends to perform poorly or equally when the padding size is large. Overall we do not observe any significant performance drop for both models as the padding size increases. This suggests that NTI learns to ignore the special padding character while processing padded sentences. The same scenario was also observed while analyzing attention weights. The attention over the padded nodes was nearly zero.

6 Discussion and Conclusion

We introduced Neural Tree Indexers, a class of tree structured recursive neural network. The NTI models achieved state-of-the-art performance on different NLP tasks. Most of the NTI models form deep neural networks and we think this is one reason that NTI works well even if it lacks direct linguistic motivations followed by other syntactic-tree-structured recursive models (Socher et al., 2013).

CNN and NTI are topologically related (Kalchbrenner and Blunsom, 2013). Both NTI and CNNs are hierarchical. However, current implementation of NTI only operates on non-overlapping subtrees while CNNs can slide over the input to produce higher-level representations. NTI is flexible in selecting the node function and the attention mechanism. Like CNN, the computation in the same tree-depth can be parallelized effectively; and therefore NTI is scalable and suitable for large-scale sequence processing. Note that NTI can be seen as a generalization of LSTM. If we construct left-branching trees in a bottom-up fashion, the model acts just like sequential LSTM. Different branching factors for the underlying tree structure have yet to be explored. NTI can be extended so it learns to select and compose dynamic number of nodes for efficiency, essentially discovering intrinsic hierarchical structure in the input.

Acknowledgments

We would like to thank the anonymous reviewers for their insightful comments and suggestions. This work was supported in part by the grant HL125089 from the National Institutes of Health (NIH). Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR 2015*.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *Neural Networks, IEEE Transactions on*, 5(2):157–166.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015a. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal, September. Association for Computational Linguistics.

Samuel R. Bowman, Christopher D. Manning, and Christopher Potts. 2015b. Tree-structured composition in neural networks without tree-structured architectures. In *Proceedings of the 2015 NIPS Workshop on Cognitive Computation: Integrating Neural and Symbolic Approaches-Volume 1583*, pages 37–42.

Samuel R. Bowman, Jon Gauthier, Abhinav Rastogi, Raghav Gupta, Christopher D. Manning, and Christopher Potts. 2016. A fast unified model for parsing and sentence understanding. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1466–1477, Berlin, Germany, August. Association for Computational Linguistics.

Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long short-term memory-networks for machine reading. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 551–561, Austin, Texas, November. Association for Computational Linguistics.

Jeffrey L Elman. 1990. Finding structure in time. *Cognitive science*, 14(2):179–211.

Christoph Goller and Andreas Kuchler. 1996. Learning task-dependent distributed representations by backpropagation through structure. In *Neural Networks, 1996., IEEE International Conference on*, volume 1, pages 347–352. IEEE.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *NIPS 2015*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Sepp Hochreiter. 1998. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116.

Ozan Irsoy and Claire Cardie. 2015. Modeling compositionality with multiplicative recurrent neural networks. In *ICLR 2015*.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709. Association for Computational Linguistics.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar, October. Association for Computational Linguistics.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *ICLR 2014*.

Diederik P Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *ICLR 2014*.

Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan Gulrajani, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *Proceedings of The 33rd International Conference on Machine Learning (ICML 2016)*, pages 1378–1387, New York, NY, USA, June.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California, June. Association for Computational Linguistics.

Quoc V Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *ICML 2014*, volume 14, pages 1188–1196.

Yishu Miao, Lei Yu, and Phil Blunsom. 2016. Neural variational inference for text processing. In *ICLR 2016*.

Lili Mou, Rui Men, Ge Li, Yan Xu, Lu Zhang, Rui Yan, and Zhi Jin. 2016. Natural language inference by tree-based convolution and heuristic matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 130–136, Berlin, Germany, August. Association for Computational Linguistics.

Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2249–2255, Austin, Texas, November. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October. Association for Computational Linguistics.

Jordan B. Pollack. 1990. Recursive distributed representations. *Artificial Intelligence*, 46(1):77–105.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. 2016. Reasoning about entailment with neural attention. In *ICLR 2016*.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 151–161, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211, Jeju Island, Korea, July. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA, October. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*, pages 3104–3112.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory

networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566, Beijing, China, July. Association for Computational Linguistics.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *NIPS 2015*.

Shuohang Wang and Jing Jiang. 2016. Learning natural language inference with lstm. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1442–1451, San Diego, California, June. Association for Computational Linguistics.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2013–2018, Lisbon, Portugal, September. Association for Computational Linguistics.

Wen-tau Yih, Ming-Wei Chang, Christopher Meek, and Andrzej Pastusiak. 2013. Question answering using enhanced lexical semantic models. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1744–1753, Sofia, Bulgaria, August. Association for Computational Linguistics.

Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. 2014. Deep learning for answer sentence selection. In *NIPS Deep Learning Workshop 2014*.

Xiao-Dan Zhu, Parinaz Sobhani, and Hongyu Guo. 2015. Long short-term memory over recursive structures. In *ICML*, pages 1604–1612.