Tag-Enhanced Dynamic Compositional Neural Network over Arbitrary Tree Structure for Sentence Representation

Chunlin Xu[a], Hui Wang[a,*], Shengli Wu[a], Zhiwei Lin[b]

[a]*Faculty of Computing, Engineering and Built Environment, Ulster University, Belfast, UK*
[b]*School of Mathematics and Physics, Queen's University, Belfast, UK*

[*]Corresponding author
Email addresses: xu-c@ulster.ac.uk (Chunlin Xu), h.wang@ulster.ac.uk (Hui Wang), s.wu1@ulster.ac.uk (Shengli Wu), Z.Lin@qub.ac.uk (Zhiwei Lin)

Abstract

Learning the distributed representation of a sentence is a fundamental operation for a variety of natural language processing tasks, such as text classification, machine translation, and text semantic matching. Tree-structured dynamic compositional networks have achieved promising performance in sentence representation due to its ability in capturing the richness of compositionality. However, existing dynamic compositional networks are mostly based on binarized constituency trees which cannot represent the inherent structural information of sentences effectively. Moreover, syntactic tag information, which is demonstrated to be useful in sentence representation, has been rarely exploited in existing dynamic compositional models. In this paper, a novel LSTM structure, ARTree-LSTM, is proposed to handle general constituency trees in which each non-leaf node can have any number of child nodes. Based on ARTree-LSTM, a novel network model, *Tag-Enhanced Dynamic Compositional Neural Network* (TE-DCNN), is proposed for sentence representation learning, which contains two ARTree-LSTMs, i.e. tag-level ARTree-LSTM and word-level ARTree-LSTM. The tag-level ARTree-LSTM guides the word-level ARTree-LSTM in conducting dynamic composition. Extensive experiments demonstrate that the proposed TE-DCNN achieves state-of-the-art performance on text classification and text semantic matching tasks.

Keywords: Sentence representation, Text classification, Text semantic matching, Dynamic compositional networks

1. Introduction

Representing sentences in terms of compact semantic vectors is an important operation for various natural language processing (NLP) tasks, such as text classification (Li & Roth, 2002; Zhu et al., 2015), text semantic matching (Wang et al., 2017b; Xu et al., 2019) and machine translation (Cho et al., 2014). Existing deep neural networks for sentence representation can be generally grouped into four types: sequence models, convolutional models, self-attention models and recursive models. The most widely used sequence models are recurrent neural network (RNN) (Elman, 1990) and its variants such as Gated Recurrent Unit (GRU) (Cho et al., 2014) and Long Short-Term Memory (LSTM) (Hochreiter & Schmidhuber, 1997). Apart from sequence models, other neural architectures such as self-attention models (Shen et al., 2018) and convolutional neural networks (CNN) (Kim, 2014) are also used for sentence representation. However, these three types of models handle a sentence in a flat sequential way without considering the compositional structure of a sentence. In fact, the inherent compositional structure of a sentence, which can be described by a constituency tree or a dependency tree, is an integral part of its meaning (Kim et al., 2019). Thus, recursive neural network (RecNN) or

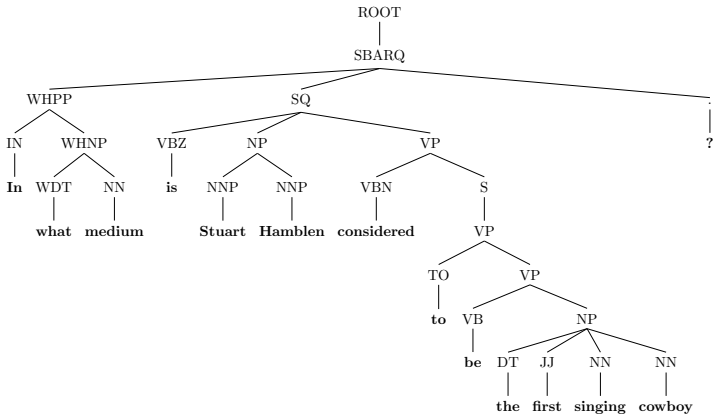
tree-structured neural network[1] models (Socher et al., 2011, 2013b) are devised to obtain the structural information of a sentence explicitly. RecNN models have achieved impressive results for sentence representation on several downstream NLP tasks such as text classification, text semantic matching and so on (Tai et al., 2015; Yang et al., 2017; Cheng et al., 2018). However, a major limitation is that all kinds of syntactic compositions share the same parameters in RecNN models, neglecting the fact that different compositions require different composition rules. For example, the *adjective-noun* composition is significantly different from the *adverb-adjective* composition. Thus, some dynamic compositional models have been proposed to model the diversity of different syntactic compositions (Hashimoto et al., 2013; Dong et al., 2014; Qian et al., 2015; Wang et al., 2017a; Liu et al., 2017c; Huang et al., 2017; Kim et al., 2019; Shen et al., 2020). For example, Shen et al. (2020) proposes a hypernetwork RecNN which employs Part-of-Speech (POS) tag[2] information and semantic information of word/phrase jointly as inputs to generate parameters of different syntactic compositions dynamically, and achieves impressive performance on text classification and text semantic matching.

In spite of the impressive performance, there are at least two limitations with previous dynamic compositional models. The first limitation is that existing dynamic compositional models are based on a binarized constituency tree which is different from the original constituency tree generated by a parser and cannot obtain the inherent structural information of a sentence effectively. Figure 1 shows a binarized constituency tree and the original constituency tree for a sentence from the TREC dataset (Li & Roth, 2002). It is clear that, different from the binarized constituency tree, the number of child nodes of a non-leaf node of the original constituency tree varies significantly. Furthermore, the original constituency tree is more shallow and so can represent a sentence in a more compact format than its corresponding binarized tree. In addition, recently proposed TreeNet (Cheng et al., 2018) has also demonstrated the effectiveness of using the original constituency tree for sentence representation. However, similar to early RecNN models, TreeNet cannot capture the diversity of different syntactic compositions.

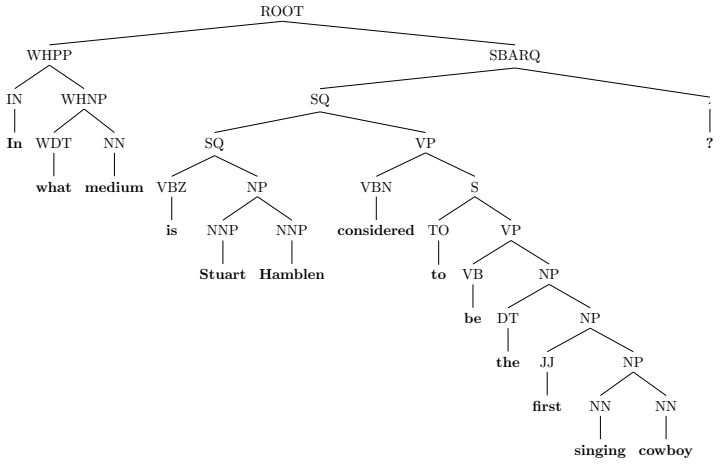
The second limitation is that previous models only use the tag embedding of the current phrase whilst composing the representation of the phrase, which is not reasonable in some cases. For example, in Figure 1 (a), although phrase *"Stuart/NNP Hamblen/NNP"* and phrase *"the/DT first/JJ singing/NN cowboy/NN"* both have the same NP tag, these two phrases have different internal structure because they vary greatly in size and the tags of their child nodes are quite different. Therefore, it is not reasonable to guide the composition of these two phrases solely based on the simple tag embedding of the NP tag. Thus, more complicated tag representations are needed to obtain structural

[1]Recursive neural network or tree-structured neural network is denoted as RecNN in this paper to distinguish them from recurrent neural network (RNN).

[2]For simplicity, we refer to Part-of-Speech (POS) tags as tags in the rest of the paper.



(a) Original constituency tree



(b) Binarized constituency tree

Figure 1: Original constituency tree and binarized constituency tree for sentence "In what medium is Stuart Hamblen considered to be the first singing cowboy ?".

information. Moreover, tag representations of child nodes are important for distinguishing different syntactic compositions.

To alleviate the above two limitations, we firstly propose a variant of LSTM, ARTree-LSTM, to handle arbitrary tree structure, in which each node within a tree can learn from its left sibling and right most child from left to right and bottom to top directions. Based on ARTree-LSTM, *Tag-Enhanced Dynamic Compositional Neural Network* (TE-DCNN) is proposed for sentence representation learning, which contains two ARTree-LSTMs, one tag-level ARTree-LSTM and one word-level ARTree-LSTM. The tag-level ARTree-LSTM accepts original tag

embedding at each node of the constituency tree and outputs the computed tag representations for the nodes, and then these tag representations will be used to control the gates of the word-level ARTree-LSTM to conduct dynamic semantic composition. For word-level ARTree-LSTM, a gate-memory encoder (Cheng et al., 2018) is used to obtain the word representation of each leaf node in the constituency tree. For each non-leaf nodes, its representation is computed based on the states and tag representations of its left sibling and right most child. The final representation of the root node is used as sentence representation. To the best of our knowledge, the proposed TE-DCNN is the first work that has the ability of both handling arbitrary tree structures such as the original constituency tree and capturing the richness of compositionality.

This paper is structured as follows. Related works are discussed in Section 2. ARTree-LSTM is presented in Section 3, and the complete model is presented in Section 4. Section 5 describes two applications of the proposed model. Experimental results on text classification and text semantic matching are given in Section 6. Ablation study is shown in Section 7. Section 8 presents error analysis and Section 9 concludes this paper.

2. Related Works

2.1. Recursive Neural Networks for Sentence Representation

Recursive neural networks (RecNNs) or tree-structured neural networks are a type of neural architecture which learns sentence representation based on recursive syntactic structures. Given a syntactic parse tree, RecNNs convert each word at the leaf nodes of the tree to a representation vector firstly, and then a compositional function such as a feed-forward neural network, is used on the representation vectors of word/phrase pairs to get the representations of non-leaf nodes recursively over trees. Finally, the representation of the root node is used as sentence representation. The earliest RecNN model was proposed by Socher et al. (2011), in which a simple feed-forward neural network is used as the compositional function of the model. To improve the performance of the basic RecNN model, researchers have developed some more effective compositional functions instead of feed-forward neural network. Socher et al. (2012) used matrix-vector multiplication as compositional function in order to model semantic composition adaptively. Socher et al. (2013b) utilizes more complex tensor computation as compositional function. However, these RecNN models suffers from the vanishing gradient problem over deep structures, resulting in difficulties in capturing long-distance dependencies in the structures. Therefore, Tree-LSTM (Tai et al., 2015) has been proposed. Tree-LSTM is a generalization of LSTM to tree-structured network topologies, which can keep long-term memory due to the well-designed gate mechanism. However, these models use the same compositional function for all kinds of syntactic compositions and so lack ability of handling dynamic compositionality for different syntactic configurations.

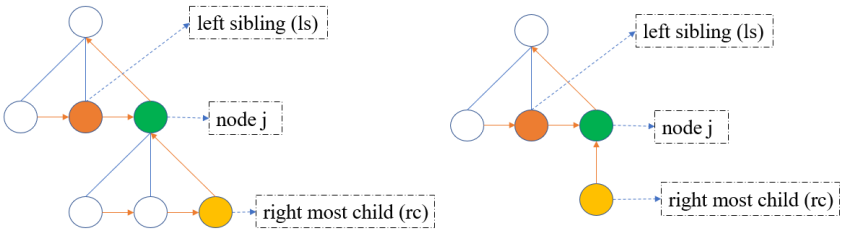
2.2. Dynamic Compositionality in Recursive Neural Networks

To perform dynamic composition, some previous works utilize multiple compositional functions, which are pre-defined according to some specific partition criterion (Socher et al., 2013a; Dong et al., 2014). Socher et al. (2013a) defined several compositional functions according to syntactic categories. For each word/phrase pair, a suitable compositional function is selected according to its syntactic categories. Dong et al. (2014) introduced AdaMC-RNTN, which also uses multiple compositional functions and adaptively chooses them based on tags and child vectors. It is difficult to predefine a general criterion that can cover all the compositional rules. Some researchers leverage tag embeddings as additional inputs for the existing tree-structured models (Qian et al., 2015; Wang et al., 2017a; Huang et al., 2017; Kim et al., 2019). For example, Wang et al. (2017a) and Huang et al. (2017) utilize tag embeddings to control the gates of the Tree-LSTM to conduct dynamic composition. Liu et al. (2017c) took advantage of hypernetwork and dynamic parameter prediction (Bertinetto et al., 2016), and proposed DC-TreeLSTM. DC-TreeLSTM composed of two separate Tree-LSTMs that have similar structure but different number of parameters. The smaller Tree-LSTM is utilized to calculate the weights of the bigger Tree-LSTM. Shen et al. (2020) proposed TG-HTreeLSTM, which improves the performance of DC-TreeLSTM by using tag embeddings as supplementary inputs for hypernetwork Tree-LSTM.

3. ARTree-LSTM

LSTM (Hochreiter & Schmidhuber, 1997) is proposed to deal with the vanishing and exploding gradient problems of RNN (Elman, 1990), which can capture long-distance dependencies for sequential data due to its well-designed gate mechanism. Tai et al. (2015) applied LSTM unit into tree structures and achieved impressive performance for sentence representation. Previous dynamic compositional models usually assume that there are only two child nodes for non-leaf nodes in Tree-LSTM, thus it can only handle a binarized constituency tree which is different from the original constituency tree generated by a standard parser tool, and cannot represent the inherent structure of a sentence effectively. Inspired by TreeNet (Cheng et al., 2018), in which each non-leaf node can learn from its left sibling and right most child, we extend the original Tree-LSTM to ARTree-LSTM to handle the original constituency tree, in which the number of child nodes of non-leaf nodes is arbitrary. Instead of learning from left and right child nodes as in Tree-LSTM, each node in ARTree-LSTM will learn from its left sibling and right most child. Child nodes with the same parent node are processed sequentially from left to right in a recurrent manner, thus the right most child can learn from all its siblings. Figure 2 shows the left sibling and right most child of a node j in two different cases. For each node j, the left sibling can be seen as its previous state and the right most child can be viewed as the representation of its descendants.

Figure 3 shows the architecture of ARTree-LSTM unit. For each node j in a constituency tree of a sentence, let $\mathbf{x}_j = [x_1, \ldots, x_{d_e}]^T$ be an input vector,



(a) node j with multiple child nodes (b) node j with only one child node

Figure 2: Left sibling and right most child of a node j in two different cases. If node j has multiple child nodes, then the last child is regarded as right most child. Otherwise, the only child is regarded as right most child.

$\mathbf{h}_j^{ls} = [h_1^{ls}, \ldots, h_d^{ls}]^T$ and $\mathbf{h}_j^{rc} = [h_1^{rc}, \ldots, h_d^{rc}]^T$ be the hidden states of left sibling and right most child of node j, respectively. $\mathbf{c}_j^{ls} = [c_1^{ls}, \ldots, c_d^{ls}]^T$ and $\mathbf{c}_j^{rc} = [c_1^{rc}, \ldots, c_d^{rc}]^T$ be the memory cells of left sibling and right most child of node j, respectively.

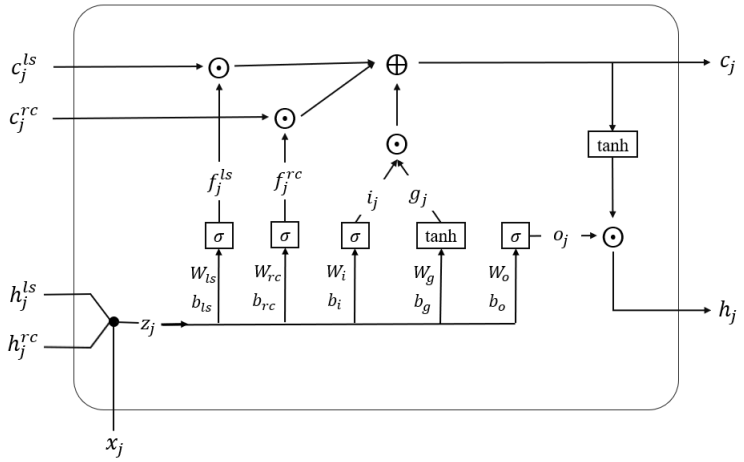


Figure 3: Illustration of ARTree-LSTM architecture.

A new vector $\mathbf{z}_j$ is created by concatenating $\mathbf{x}_j, \mathbf{h}_j^{ls}, \mathbf{h}_j^{rc}$ where $\mathbf{z}_j = [x_1, \ldots, x_{d_e}, h_1^{ls}, \ldots, h_d^{ls}, h_1^{rc}, \ldots, h_d^{rc}]^T$. Then the hidden state of each node j in ARTree-LSTM is computed by the following equations:

$$\mathbf{i}_j = \sigma(\mathbf{W}_i \mathbf{z}_j + \mathbf{b}_i) \tag{1}$$

$$\mathbf{f}_j^{ls} = \sigma(\mathbf{W}_{ls} \mathbf{z}_j + \mathbf{b}_{ls}) \tag{2}$$

$$\mathbf{f}_j^{rc} = \sigma(\mathbf{W}_{rc}\mathbf{z}_j + \mathbf{b}_{rc}) \tag{3}$$

$$\mathbf{g}_j = \tanh(\mathbf{W}_g\mathbf{z}_j + \mathbf{b}_g) \tag{4}$$

$$\mathbf{o}_j = \sigma(\mathbf{W}_o\mathbf{z}_j + \mathbf{b}_o) \tag{5}$$

$$\mathbf{c}_j = \mathbf{f}_j^{ls} \odot \mathbf{c}_j^{ls} + \mathbf{f}_j^{rc} \odot \mathbf{c}_j^{rc} + \mathbf{i}_j \odot \mathbf{g}_j \tag{6}$$

$$\mathbf{h}_j = \mathbf{o}_j \odot \tanh(\mathbf{c}_j) \tag{7}$$

where $\mathbf{c}_j, \mathbf{h}_j \in \mathbb{R}^d$ refer to the memory cell and hidden state of node j. $\mathbf{i}_j, \mathbf{f}_j^{ls}, \mathbf{f}_j^{rc}$, $\mathbf{o}_j \in \mathbb{R}^d$ represent input gate, two forgot gates (left sibling and right most child), and output gate, respectively. $\mathbf{g}_j \in \mathbb{R}^d$ is the newly composed input for the memory cell. $\mathbf{W}_i, \mathbf{W}_{ls}, \mathbf{W}_{rc}, \mathbf{W}_g, \mathbf{W}_o \in \mathbb{R}^{d\times(2d+d_e)}$ and $\mathbf{b}_i, \mathbf{b}_{ls}, \mathbf{b}_{rc}, \mathbf{b}_g, \mathbf{b}_o \in \mathbb{R}^d$ are trainable parameters. tanh is the hyperbolic tangent, σ denotes the sigmoid function, and $\odot$ represents element-wise multiplication.

In particular, child nodes with the same parent node are processed sequentially from left to right. If node j is a leaf node (has no child nodes), a zero vector is employed to initialize $\mathbf{h}_j^{rc}$ and $\mathbf{c}_j^{rc}$. If node j is the first child node (has no left sibling), a zero vector is used to initialize $\mathbf{h}_j^{ls}$ and $\mathbf{c}_j^{ls}$.

For simplicity, we describe the computation of the hidden state of node j at a high level with Equation (8) to facilitate references later in the paper, and the detailed computation refers to Equations (1-7).

$$\mathbf{h}_j = \mathbf{ARTree-LSTM}(\mathbf{x}_j, \mathbf{h}_j^{ls}, \mathbf{h}_j^{rc}, \mathbf{c}_j^{ls}, \mathbf{c}_j^{rc}) \tag{8}$$

All notations in Equation (8) follow equations Equations (1-7).

4. Model

In this section, a novel dynamic compositional neural architecture, named TE-DCNN (**T**ag-**E**nhanced **D**ynamic **C**ompositional **N**eural **N**etwork) is presented. The proposed TE-DCNN is composed of two separate ARTree-LSTMs: Tag-level ARTree-LSTM and word-level ARTree-LSTM. The tag-level ARTree-LSTM is employed to guide the composition of word-level ARTree-LSTM which is responsible for constructing sentence representations. For a node j in the constituency tree, instead of using its own tag representation, tag representations of its left sibling and right most child are jointly used to control the gate functions of the word-level ARTree-LSTM to perform dynamic composition. Figure 4 illustrates the proposed model.

Formally, we denote sentence as a sequence of words $(w_1, w_2, ..., w_m)$ where m is the length of the sentence, and the word embeddings as $(\mathbf{v}_1, \dots \mathbf{v}_m)$ where $\mathbf{v}_i = [v_1, \dots, v_{d_w}]^T, i \in [1, m]$. Tag embedding for the tag attached to each node

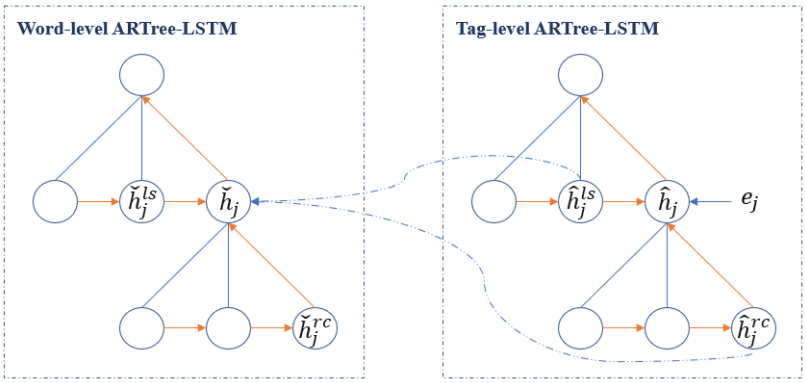


Figure 4: An overview of TE-DCNN at node j. Tag-level ARTree-LSTM only accepts tag embeddings as inputs, the hidden state $\hat{\mathbf{h}}_j$ of node j is computed based on its tag embeddings $\mathbf{e}_j$ and the hidden states of its right most child $\hat{\mathbf{h}}_j^{rc}$ and left sibling $\hat{\mathbf{h}}_j^{ls}$. Word-level ARTree-LSTM accepts word representations and tag representations generated by tag-level ARTree-LSTM as inputs, the hidden state $\check{\mathbf{h}}_j$ is computed based on hidden states of its right most child $\check{\mathbf{h}}_j^{rc}$ and left sibling $\check{\mathbf{h}}_j^{ls}$ and the corresponding tag presentations $\hat{\mathbf{h}}_j^{rc}$ and $\hat{\mathbf{h}}_j^{ls}$.

j in the constituency tree is denoted by $\mathbf{e}_j = [e_1, \ldots, e_{d_t}]^T$. For each word w_t ($t \in [1, m]$) within a sentence, we use a gate-memory encoder (Cheng et al., 2018) on its word embedding $\mathbf{v}_t$ and tag embedding $\mathbf{e}_t$ to obtain a new word representation $\bar{\mathbf{h}}_t$ by the following equations:

$$\bar{\mathbf{i}}_t = \sigma(\mathbf{U}_{vi}\mathbf{v}_t + \mathbf{U}_{ei}\mathbf{e}_t + \mathbf{a}_i) \tag{9}$$

$$\bar{\mathbf{o}}_t = \sigma(\mathbf{U}_{vo}\mathbf{v}_t + \mathbf{U}_{eo}\mathbf{e}_t + \mathbf{a}_o) \tag{10}$$

$$\bar{\mathbf{g}}_t = \sigma(\mathbf{U}_{vg}\mathbf{v}_t + \mathbf{U}_{eg}\mathbf{e}_t + \mathbf{a}_g) \tag{11}$$

$$\bar{\mathbf{c}}_t = \bar{\mathbf{i}}_t \odot \bar{\mathbf{g}}_t \tag{12}$$

$$\bar{\mathbf{h}}_t = \bar{\mathbf{o}}_t \odot \tanh(\bar{\mathbf{c}}_t) \tag{13}$$

where $\bar{\mathbf{h}}_t, \bar{\mathbf{c}}_t \in \mathbb{R}^{d_h}$ represent the hidden state and memory cell of the gate-memory encoder. $\bar{\mathbf{i}}_t, \bar{\mathbf{o}}_t \in \mathbb{R}^{d_h}$ are two gates of the gate-memory encoder. $\mathbf{U}_{vi}, \mathbf{U}_{ei}, \mathbf{U}_{vo}, \mathbf{U}_{eo}, \mathbf{U}_{vg}, \mathbf{U}_{eg} \in \mathbb{R}^{d_h \times (d_w + d_t)}$ and $\mathbf{a}_i, \mathbf{a}_o, \mathbf{a}_g \in \mathbb{R}^{d_h}$ are trainable parameters. The rest of the notations follow that of the ARTree-LSTM above. The representations $\bar{\mathbf{h}}_t$ of words in the sentence will be used as the inputs of the word-level ARTree-LSTM.

Then, the final representation for the input sentence can be obtained in two steps:

First, the tag-level ARTree-LSTM which only accepts tag embeddings as inputs is employed to obtain the structure-aware tag representations (the right side of Figure 4). Tags are usually represented as low-dimensional tag embeddings in most previous works. Kim et al. (2019) used a Tree-LSTM on original tag embeddings to obtain the structure-aware tag representations which have proven to be more effective than simple tag embeddings. Similarly, we use a separate ARTree-LSTM to obtain the structure-aware tag representation $\hat{\mathbf{h}}_j \in \mathbb{R}^{d_h}$ for each node j in a constituency tree in the following way:

$$\hat{\mathbf{h}}_j = \mathbf{ARTree-LSTM}(\mathbf{e}_j, \hat{\mathbf{h}}_j^{ls}, \hat{\mathbf{h}}_j^{rc}, \hat{\mathbf{c}}_j^{ls}, \hat{\mathbf{c}}_j^{rc}) \tag{14}$$

where $\mathbf{ARTree-LSTM}$ refers to Equation (8). $\mathbf{e}_j \in \mathbb{R}^{d_t}$ is tag embedding for the tag attached to each node j. $\hat{\mathbf{h}}_j^{ls}, \hat{\mathbf{h}}_j^{rc}, \hat{\mathbf{c}}_j^{ls}, \hat{\mathbf{c}}_j^{rc} \in \mathbb{R}^{d_h}$ represent the hidden states and memory cells of the left sibling and right most child of node j. The new tag representation $\hat{\mathbf{h}}_j$ will be used to control the gates of the word-level ARTree-LSTM to conduct dynamic composition.

Second, the word-level ARTree-LSTM (the left side of Figure 4) which accepts word representations as inputs is used to obtain the final sentence representation. Note that for each non-leaf nodes, its representation is computed based on the states and tag representations of its left sibling and right most child. Let $\check{\mathbf{h}}_j^{ls} = [\check{h}_1^{ls}, \ldots, \check{h}_{d_h}^{ls}]^T, \check{\mathbf{c}}_j^{ls} = [\check{c}_1^{ls}, \ldots, \check{c}_{d_h}^{ls}]^T, \check{\mathbf{h}}_j^{rc} = [\check{h}_1^{rc}, \ldots, \check{h}_{d_h}^{rc}]^T, \check{\mathbf{c}}_j^{rc} = [\check{c}_1^{rc}, \ldots, \check{c}_{d_h}^{rc}]^T$ be the hidden states and memory cells of left sibling and right most child of node j in the word-level ARTree-LSTM, respectively. $\bar{\mathbf{h}}_j = [\bar{h}_1, \ldots, \bar{h}_{d_h}]^T$ is the input vector of node j, which is a zero vector at non-leaf node and word representation vector computed by Equations (9-13) at leaf nodes. $\hat{\mathbf{h}}_j^{ls} = [\hat{h}_1^{ls}, \ldots, \hat{h}_{d_h}^{ls}]^T, \hat{\mathbf{h}}_j^{rc} = [\hat{h}_1^{rc}, \ldots, \hat{h}_{d_h}^{rc}]^T$ are the tag representations of the left sibling and right most child of node j, which are computed by Equation (14). Vectors $\mathbf{p}_j \in \mathbb{R}^{5d_h}$ and $\mathbf{q}_j \in \mathbb{R}^{3d_h}$ are created by concatenating $(\bar{\mathbf{h}}_j, \check{\mathbf{h}}_j^{ls}, \check{\mathbf{h}}_j^{rc}, \hat{\mathbf{h}}_j^{ls}, \hat{\mathbf{h}}_j^{ls})$ and $(\bar{\mathbf{h}}_j, \check{\mathbf{h}}_j^{ls}, \check{\mathbf{h}}_j^{rc})$, respectively. The hidden states of each node j in word-level ARTree-LSTM is defined by the following:

$$\check{\mathbf{i}}_j = \sigma(\mathbf{V}_i \mathbf{p}_j + \mathbf{r}_i) \tag{15}$$

$$\check{\mathbf{f}}_j^{ls} = \sigma(\mathbf{V}_{ls} \mathbf{p}_j + \mathbf{r}_{ls}) \tag{16}$$

$$\check{\mathbf{f}}_j^{rc} = \sigma(\mathbf{V}_{rc} \mathbf{p}_j + \mathbf{r}_{rc}) \tag{17}$$

$$\check{\mathbf{g}}_j = \tanh(\mathbf{V}_g \mathbf{q}_j + \mathbf{r}_g) \tag{18}$$

$$\check{\mathbf{o}}_j = \sigma(\mathbf{V}_o \mathbf{p}_j + \mathbf{r}_o) \tag{19}$$

$$\check{\mathbf{c}}_j = \check{\mathbf{f}}_j^{ls} \odot \check{\mathbf{c}}_j^{ls} + \check{\mathbf{f}}_j^{rc} \odot \check{\mathbf{c}}_j^{rc} + \check{\mathbf{i}}_j \odot \check{\mathbf{g}}_j \tag{20}$$

$$\check{\mathbf{h}}_j = \check{\mathbf{o}}_j \odot \tanh(\check{\mathbf{c}}_j) \tag{21}$$

where $\check{\mathbf{h}}_j, \check{\mathbf{c}}_j \in \mathbb{R}^{d_h}$ refer to the hidden state and memory cell of node j in the word-level ARTree-LSTM. $\mathbf{V}_i, \mathbf{V}_{ls}, \mathbf{V}_{rc}, \mathbf{V}_o \in \mathbb{R}^{d_h \times 5d_h}, \mathbf{V}_g \in \mathbb{R}^{d_h \times 3d_h}$ and $\mathbf{r}_i, \mathbf{r}_{ls}, \mathbf{r}_{rc}, \mathbf{r}_g, \mathbf{r}_o \in \mathbb{R}^{d_h}$ are learnable parameters. The remaining notation follows Equations (1-7). Finally, the hidden state of the root node $\check{\mathbf{h}}^{\mathbf{root}} \in \mathbb{R}^{d_h}$ in the word-level ARTree-LSTM is used as the representation for the given sentence.

5. Applications of TE-DCNN

In this section, we describe the applications of TE-DCNN for two typical NLP tasks.

Text classification. Given a sentence s and a pre-defined class set $\mathcal{Y}$, text classification is to predict a label $\hat{y} \in \mathcal{Y}$ for s. A softmax classifier is applied directly on the sentence representation $\check{\mathbf{h}}^{\mathbf{root}} \in \mathbb{R}^{d_h}$ in this paper. The final predicted probability distribution of class y given sentence s is defined as following:

$$p(y|s) = \mathbf{softmax}(\mathbf{W}_c \check{\mathbf{h}}^{\mathbf{root}} + \mathbf{b_c}) \tag{22}$$

where $\mathbf{W}_c \in \mathbb{R}^{d_c \times d_h}, \mathbf{b_c} \in \mathbb{R}^{d_c}$ are trainable parameters, d_c is the size of class set $\mathcal{Y}$.

Text semantic matching. Text semantic matching is to predict a label $\hat{l}$ which represents the relationship between a given sentence pair $s1$ and $s2$ from a pre-defined label set $\mathcal{L}$. Firstly, the same TE-DCNN is used to encode $s1$ and $s2$ into two sentence representation vectors $\check{\mathbf{h}}^{\mathbf{root}}_{\mathbf{s1}}, \check{\mathbf{h}}^{\mathbf{root}}_{\mathbf{s2}} \in \mathbb{R}^{d_h}$. Next, some matching heuristics (Mou et al., 2016) are used to combine these two sentence representation vectors together in the following way:

$$\mathbf{h}_{st} = [\check{\mathbf{h}}^{root}_{s1}, \check{\mathbf{h}}^{root}_{s2}, \check{\mathbf{h}}^{root}_{s1} \odot \check{\mathbf{h}}^{root}_{s2}, |\check{\mathbf{h}}^{root}_{s1} - \check{\mathbf{h}}^{root}_{s2}|] \tag{23}$$

Then, a single layer MLP with ReLU activation function is applied on the above concatenated vector $\mathbf{h}_{st}$:

$$\mathbf{h}_m = \mathbf{Relu}(\mathbf{W}_m \mathbf{h}_{st} + \mathbf{b}_m) \tag{24}$$

where $\mathbf{h}_m \in \mathbb{R}^{d_m}$ is the intermediate feature vector for the following classifier. $\mathbf{W}_m \in \mathbb{R}^{d_m \times 4d_h}, \mathbf{b}_m \in \mathbb{R}^{d_m}$ are trainable parameters. Finally, the probability distribution of label l given sentence pair $s1$ and $s2$ is obtained using a softmax classifier:

$$p(l|(s1, s2)) = \mathbf{softmax}(\mathbf{W}_l \mathbf{h}_m + \mathbf{b}_l) \tag{25}$$

where $\mathbf{W}_l \in \mathbb{R}^{d_l \times d_m}, \mathbf{b}_l \in \mathbb{R}^{d_l}$ are again trainable parameters. The parameters of the model are learned to minimise the cross-entropy of the distributions between predicted and true label.

6. Experiments

6.1. Datasets

We evaluate the proposed model on three benchmarks for text classification (MR, SUBJ, TREC) and one dataset (SICK) for text semantic matching:

- MR: The movie reviews with positive or negative label (Pang & Lee, 2005)[3].
- SUBJ: Sentences grouped as being either subjective or objective (Pang & Lee, 2004)[4].
- TREC: A dataset which groups questions into six different question types (Li & Roth, 2002)[5].
- SICK: A textual entailment dataset with three classes (entailment, neutral, contradiction) (Marelli et al., 2014)[6].

Tabel 1 shows the detailed statistics about the above four datasets.

Table 1: Statistics of four benchmark datasets for two tasks. Train, Dev and Test are the size of train, validation and test dataset, respectively. CV means 10-fold cross validation is used. L_{avg} is the average number of words in sentences. $|V|$ is the size of vocabulary. $|T|$ is the number of tags. Class is the size of label/class set.

Dataset	**Train**	**Dev**	**Test**	$\mathbf{L_{avg}}$	**|V|**	**|T|**	**Class**
MR	10662	-	CV	22	19K	73	2
SUBJ	10000	-	CV	21	21K	72	2
TREC	75952	-	500	10	10K	67	6
SICK	4500	500	4927	10	2K	41	3

6.2. Experimental Setup

In all experiments, sentences in datasets are tokenized and parsed by Stanford Stanford PCFG parser[7] (Klein & Manning, 2003). Word embeddings are initialized with the 300-dimensional GloVe word vectors (Pennington et al., 2014), and out-of-vocabulary words are randomly sampled from the uniform distribution $[-0.05, 0.05]$. Tag embeddings are initialized by the normal distribution $[0, 1]$. Tag embeddings are fine-tuned during training procedure while word embeddings are fixed. Hidden size for ARTree-LSTMs is fine-tuned from

[3]https://www.cs.cornell.edu/people/pabo/movie-review-data/
[4]https://www.cs.cornell.edu/people/pabo/movie-review-data/
[5]https://cogcomp.seas.upenn.edu/Data/QA/QC/
[6]https://wiki.cimec.unitn.it/tiki-index.php?page=CLIC
[7]https://nlp.stanford.edu/software/lex-parser.shtml

[100, 200, 300]. The dimension of tag embedding is selected from [20, 25]. Batch size is selected from [32, 64]. Weights of the model are trained by minimizing the cross-entropy of the training dataset by Adam optimizer (Kingma & Ba, 2014) and learning rate is fine-tuned in the range of $[1e^{-2}, 1e^{-3}]$. The accuracy metric is used in this paper to measure the performance of the proposed and all comparison models on four datasets, and the results of all comparison models come from respective papers.

6.3. Results

Text Classification The proposed TE-DCNN is compared with two types of models. The first is RecNN based models which are based on tree structure, and the other is non-RecNN based models. Table 2 and Table 3 show test accuracies of the proposed TE-DCNN and comparison models on each dataset.

Table 2: Accuracies of previous RecNN based models and the proposed TE-DCNN on three text classification datasets.

Model	**MR**	**SUBJ**	**TREC**
Non-Dynamic Models			
RecNN (Socher et al., 2011)	76.4	91.8	90.2
Tree-LSTM (Tai et al., 2015)	81.2	93.2	93.6
BiTreeLSTM (Teng & Zhang, 2017)	-	-	94.8
TreeNet (Cheng et al., 2018)	83.6	95.9	96.1
Dynamic Compositional Models			
AdaHT-LSTM (Liu et al., 2017b)	81.9	94.1	-
iTLSTM (Liu et al., 2017a)	82.5	94.5	-
DC-RecNN (Liu et al., 2017c)	80.2	93.5	91.2
DC-TreeLSTM (Liu et al., 2017c)	81.7	93.7	93.8
TE-RNN (Qian et al., 2015)	77.9	-	-
TE-LSTM (Huang et al., 2017)	82.2	-	-
SATA Tree-LSTM(Kim et al., 2019)	83.8	95.4	96.2
TG-HRecNN (Shen et al., 2020)	80.9	93.7	93.6
TG-HTreeLSTM (Shen et al., 2020)	82.6	94.9	95.8
TE-DCNN (proposed)	**84.1**	**95.1**	**97.0**

Comparison with RecNN based Models. We classified RecNN based models into two categories: Non-dynamic models and dynamic compositional models. Non-dynamic models are RecNN based models that share the same parameters for all kinds of syntactic compositions, while dynamic compositional models can distinguish different syntactic compositions. Table 2 shows test accuracies of the proposed TE-DCNN and previous RecNN based models. **Firstly**, compared with all previous RecNN based models, TE-DCNN achieves superior or competitive performance on all three text classification datasets. Moreover, it sets a new state of the art among RecNN based models on two out of

three datasets - MR and TREC with accuracies of 84.1% and 97%, respectively. Specifically, the proposed TE-DCNN outperforms the best non-dynamic model and dynamic compositional model on MR dataset by a margin of 0.5% and 0.3%; and on TREC dataset by a margin of 0.9% and 0.8%, respectively. **Secondly**, the proposed model is superior to dynamic compositional models ignoring tag information (i.e., AdaHT-LSTM, iTLSTM, DC-RecNN and DC-TreeLSTM) on all three datasets MR, SUBJ and TREC with 2.4%, 0.6% and 3.2% improvements, respectively. The consistency suggests these improvements are due to the usage of tag information which is helpful for composing sentence representation. **Thirdly**, compared with five dynamic models leveraging tag information (i.e., TE-RNN, TE-LSTM, TG-HRecNN, TG-HTreeLSTM and SATA Tree-LSTM), TE-DCNN outperforms these models on MR and TREC datasets by a margin of 0.3% and 0.8%, respectively, and achieves competitive performance with the state-of-the-art model SATA Tree-LSTM on SUBJ dataset.

Table 3: Accuracies of Non-RecNN based models and the proposed TE-DCNN on three text classification datasets. The symbol * indicates models which are pre-trained with large external corpora.

Model	**MR**	**SUBJ**	**TREC**
CNN (Kim, 2014)	81.5	93.4	93.6
BLSTM-2DCNN (Zhou et al., 2016)	82.3	94.0	96.1
byte-mLSTM* (Radford et al., 2017)	86.9	94.6	-
BCN + Char + CoVe* (McCann et al., 2017)	-	-	95.8
DARLM (Zhou et al., 2018)	83.2	94.1	96.0
3W-CNN (Zhang et al., 2019)	82.3	93.5	-
WSAN (Huang et al., 2019)	83.2	94.6	95.0
TE-DCNN (proposed)	**84.1**	**95.1**	**97.0**

Comparison with Non-RecNN based Models. Table 3 shows a comparison with some non-RecNN based models on three text classification datasets. We observe that the proposed TE-DCNN has consistently strong performance on three datasets, and it outperforms all comparison models in this group on SUBJ and TREC datasets by a margin of 0.5% and 0.9%, respectively. Although byte-mLSTM achieves the state-of-the-art performance on MR dataset with an accuracy of 86.9%, better than TE-DCNN with an accuracy of 84.1%, it is pre-trained with large external corpora while TE-DCNN do not perform any pre-train procedure.

Text Semantic Matching. To evaluate the proposed TE-DCNN on other NLP tasks, we also conducted an experiment on text semantic matching task. Different from text classification which works by classifying one sentence at a time, text semantic matching works by comparing two sentences at a time. Therefore, in a text classification dataset, a sample is a single sentence whereas in a text semantic matching dataset, a sample is a pair of sentences.Text semantic matching experiments were conducted on the SICK (Marelli et al.,

2014) dataset, and the experimental results are shown in Table 4. The results of TreeNet and SATA TreeLSTM come from our implementation while results of other comparison models come from respective papers. We can see that TE-DCNN has again demonstrated its superior performance compared against the previous RecNN based models. Specifically, TE-DCNN outperforms all compared non-dynamic models by a margin of 1.2% and is slightly better than previous dynamic compositional models with 0.1% improvements. This comparison shows that TE-DCNN is also effective in text sematic matching task, and has generalization ability in different NLP tasks.

Table 4: Accuracies of previous RecNN based models and the proposed TE-DCNN on the SICK dataset.

Model	**SICK**
Non-Dynamic Models	
RecNN(Socher et al., 2011)	74.9
MV-RNN (Socher et al., 2012)	75.5
RNTN (Socher et al., 2013b)	76.9
TreeLSTM (Tai et al., 2015)	77.5
TreeNet (Cheng et al., 2018)	82.1
Dynamic Compositional Models	
DC-RecNN (Liu et al., 2017c)	80.2
DC-TreeLSTM (Liu et al., 2017c)	82.3
TG-HRecNN (Shen et al., 2020)	77.5
TG-HTreeLSTM (Shen et al., 2020)	83.3
SATA Tree-LSTM (Kim et al., 2019)	83.3
TE-DCNN (proposed)	**83.4**

7. Ablation Study

In this section, we present an ablation study on key modules of the proposed TE-DCNN to explore their effectiveness. We focus on two modules, the ARTree-LSTM and tag representations. TREC dataset is used in this experiment, and the target module is replaced with other candidates while keeping the other settings fixed. In the first case, the ARTree-LSTM is replaced with a basic Tree-LSTM. In the second case, we do not employ any tag information in the model.

The experimental results are shown in figure 5. As the chart shows, TE-DCNN outperforms the other two options we considered. Specially, if we do not use any tag information in TE-DCNN, the accuracy of the model drops to 95.6%, with 1.4% performance degradation. A possible reason for this performance degradation is that the model cannot distinguish different syntactic compositions while neglecting tag information, because tag representations are used to control the gates of the word-level ARTree-LSTM to conduct dynamic

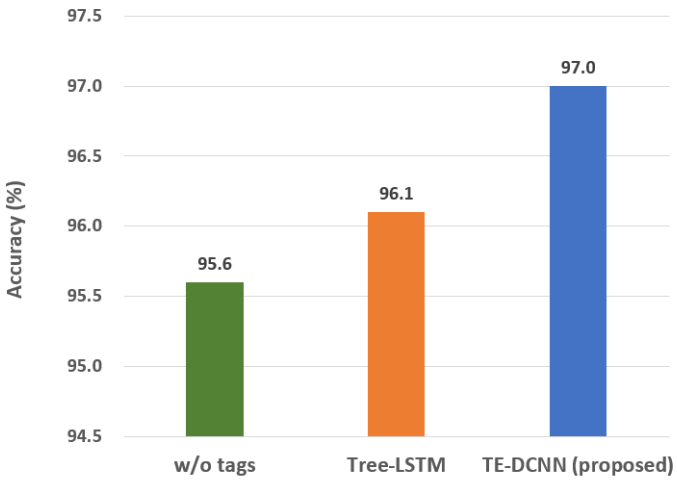


Figure 5: An ablation study on key modules of TE-DCNN. Test accuracies on TREC dataset are reported. Tree-LSTM: Tree-LSTM is used instead of ARTree-LSTM in TE-DCNN. w/o tags: Tag information is not used.

composition in TE-DCNN. If we replace ARTree-LSTM with a basic Tree-LSTM in TE-DCNN, the accuracy of the model drops to 96.1%, with 0.9% performance degradation. Possible reason for this performance degradation is because basic Tree-LSTM can only handle a binarized constituency tree which is different from the original constituency tree. In contrast, ARTree-LSTM can handle the original constituency tree of a sentence, thus can capture the inherent structural information of a sentence effectively and has better expressive ability than a basic Tree-LSTM.

8. Error Analysis

In this section, we analyze the predictions of the proposed TE-DCNN model and another two state-of-the-art models, TreeNet (Cheng et al., 2018) and SATA Tree-LSTM (Kim et al., 2019). SATA TreeLSTM and our model are dynamic compositional models while TreeNet is a non-dynamic model. Table 5 gives a breakdown of accuracy for classes on test sets of TREC and SICK datasets. We observe that, for TREC dataset, the proposed TE-DCNN matches the other two models on all classes. For SICK dataset, most of our gains stem from Neutral, while most losses come from Entailment pairs. Figure 6 presents the distribution of errors of our TE-DCNN and other two models on SICK dataset. There are six types of error in total. We can see that the most frequent error type of the proposed model is E→N which means our model tend to classify an Entailment sentence pair Neutral. While the most frequent type of error of TreeNet and

SATA TreeLSTM is N→E. For all the three models, the least frequent error type is E→C. Moreover, in most cases, dynamic compositional models such as SATA TreeLSTM and the propsoed TE-DCNN have smaller error rate than non-dynamic model TreeNet except N→C.

Table 6 shows some example wins and losses of the proposed TE-DCNN compared to other models on SICK dataset. Examples 1 and 2 are cases where the proposed TE-DCNN is correct while both TreeNet and SATA TreeLSTM are incorrect. In these two examples, both sentences contain phrases that are either the same or highly lexically related (e.g., "a dog", "toddler/baby" and "a toy/a ball"). Our TE-DCNN correctly favors Neutral in these cases, while TreeNet and SATA TreeLSTM prefer to Entailment. Due to this characteristic of TE-DCNN, it fails in Examples 3 and 4 while TreeNet and SATA TreeLSTM succeed. Examples 5 and 6 are cases where the proposed TE-DCNN and SATA TreeLSTM are correct and TreeNet is incorrect. In these two examples, the key point to predict correctly is to conclude that "a white and brown dog/no white and brown dog" and "no girl/one girl" are contradictions. A possible reason for the success of TE-DCNN and SATA TreeLSTM in these two cases is that these two models conduct dynamic composition with the aid of syntactic tags, so they can find the differences between above two phrase pairs easily.

Datasets	**Class**	**TreeNet**	**SATA Tree-LSTM**	**TE-DCNN**
TREC	NUM	94.7	95.6	**96.5**
	HUM	98.5	98.5	98.5
	ENTY	89.4	**93.6**	**93.6**
	LOC	95.1	96.3	**97.5**
	DESC	96.4	97.8	**98.6**
	ABBR	88.9	88.9	**98.6**
	Overall	96.1	96.2	**97.0**
SICK	Entailment	84	**84.9**	77.9
	Neutral	81.8	82.2	**86.2**
	Contradiction	78.4	**85.3**	83.3
	Overall	82.1	83.3	**83.4**

Table 5: Breakdown of accuracy with respect to classes on TREC and SICK test sets. TE-DCNN is the proposed model while TreeNet (Cheng et al., 2018) and SATA Tree-LSTM (Kim et al., 2019) are two previously developed models with state-of-the-art performance (see Tables 2 and 4).

9. Conclusion

In this paper, we have presented a novel dynamic compositional model that exploits tree structures for sentence representation. A newly introduced ARTree-LSTM is employed to handle the original constituency tree firstly, in which an inner node can have arbitrary child nodes. Then a tag-level ARTree-LSTM and

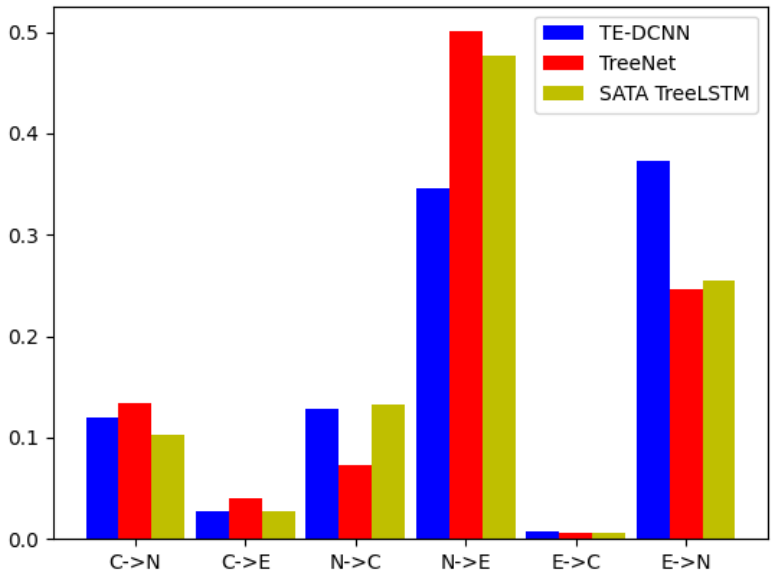


Figure 6: Distribution of errors on SICK dataset. C, N and E refer to Contradiction, Neutral and Entailment, respectively. C→N means the true label is Contradiction while the prediction is Neutral. and so on.

ID	Text pair	TreeNet	SATA	TE-DCNN	Gold
1	a dog is licking a toddler a dog is licking a baby	E	E	N	N
2	a dog is running towards a ball a dog is running through a field and is chasing a toy	E	E	N	N
3	a deer is jumping over a fence a deer is jumping over the enclosure	E	E	N	E
4	a man is playing soccer a man is playing a game with a ball	E	E	N	E
5	a white and brown dog is walking through the water with difficulty there is no white and brown dog pacing with through the water difficulty	N	C	C	C
6	there is no girl jumping into the car one girl is jumping on the car	N	C	C	C

Table 6: Example wins and losses on SICK dataset of the proposed TE-DCNN compared to two state-of-the-art models TreeNet (Cheng et al., 2018) and SATA (i.e., SATA Tree-LSTM (Kim et al., 2019).) E, C and N refer to Entailment, Contradiction and Neutral, respectively.

a word-level ARTree-LSTM are jointly used for sentence representation. Experiments on four datasets and two NLP tasks have demonstrated the superiority and the generalization ability of the proposed model. In future work, we plan to explore a new way of conducting dynamic composition over arbitrary tree structure using hypernetworks.

Acknowledgment

This work is partially funded by Ulster University VCRS scholarship and the EU Horizon 2020 research innovation programme under grant agreement No 700381 for project ASGARD (ANALYSIS SYSTEM FOR GATHERED RAW DATA).

References

Bertinetto, L., Henriques, J. F., Valmadre, J., Torr, P., & Vedaldi, A. (2016). Learning feed-forward one-shot learners. In *Advances in Neural Information Processing Systems* (pp. 523–531). Barcelona, Spain. URL: `http://papers.nips.cc/paper/6068-learning-feed-forward-one-shot-learners.pdf`.

Cheng, Z., Yuan, C., Li, J., & Yang, H. (2018). Treenet: Learning sentence representations with unconstrained tree structure. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence* (pp. 4005–4011). Stockholm, Sweden. URL: `https://www.ijcai.org/proceedings/2018/0557.pdf`.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1724–1734). Doha, Qatar. URL: `http://www.emnlp2014.org/papers/pdf/EMNLP2014179.pdf`.

Dong, L., Wei, F., Tan, C., Tang, D., Zhou, M., & Xu, K. (2014). Adaptive recursive neural network for target-dependent twitter sentiment classification. In *Proceedings of the 52nd annual meeting of the association for computational linguistics (volume 2: Short papers)* (pp. 49–54). Baltimore, Maryland. doi:`10.3115/v1/P14-2009`.

Elman, J. L. (1990). Finding structure in time. *Cognitive science*, *14*, 179–211. URL: `https://doi.org/10.1207/s15516709cog1402_1`.

Hashimoto, K., Miwa, M., Tsuruoka, Y., & Chikayama, T. (2013). Simple customization of recursive neural networks for semantic relation classification. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing* (pp. 1372–1376). Seattle, Washington, USA. URL: `https://www.aclweb.org/anthology/D13-1137`.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*, 1735–1780. doi:https://doi.org/10.1162/neco.1997.9.8.1735.

Huang, M., Qian, Q., & Zhu, X. (2017). Encoding syntactic knowledge in neural networks for sentiment classification. *ACM Transactions on Information Systems (TOIS)*, *35*, 26. doi:10.1145/3052770.

Huang, T., Deng, Z.-H., Shen, G., & Chen, X. (2019). A window-based self-attention approach for sentence encoding. *Neurocomputing*, . URL: https://doi.org/10.1016/j.neucom.2019.09.024.

Kim, T., Choi, J., Edmiston, D., Bae, S., & Lee, S.-g. (2019). Dynamic compositionality in recursive neural networks with structure-aware tag representations. In *Proceedings of the AAAI Conference on Artificial Intelligence* (pp. 6594–6601). Honolulu,Hawaii. doi:https://doi.org/10.1609/aaai.v33i01.33016594.

Kim, Y. (2014). Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1746–1751). Doha, Qatar. doi:10.3115/v1/D14-1181.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, . URL: https://arxiv.org/pdf/1412.6980.pdf.

Klein, D., & Manning, C. D. (2003). Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1* (pp. 423–430). Sapporo, Japan. doi:10.3115/1075096.1075150.

Li, X., & Roth, D. (2002). Learning question classifiers. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1* (pp. 1–7). Taipei, Taiwan. doi:10.3115/1072228.1072378.

Liu, P., Qian, K., Qiu, X., & Huang, X. (2017a). Idiom-aware compositional distributed semantics. In *Proceedings of the 2017 conference on empirical methods in natural language processing* (pp. 1204–1213). Copenhagen, Denmark. doi:10.18653/v1/D17-1124.

Liu, P., Qiu, X., & Huang, X. (2017b). Adaptive semantic compositionality for sentence modelling. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence* (pp. 4061–4067). Melbourne, Australia. URL: https://www.ijcai.org/proceedings/2017/0567.pdf.

Liu, P., Qiu, X., & Huang, X. (2017c). Dynamic compositional neural networks over tree structure. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence* (pp. 4054–4060). Melbourne, Australia. URL: https://www.ijcai.org/proceedings/2017/0566.pdf.

Marelli, M., Bentivogli, L., Baroni, M., Bernardi, R., Menini, S., & Zamparelli, R. (2014). Semeval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. In *Proceedings of the 8th international workshop on semantic evaluation (SemEval 2014)* (pp. 1–8). Dublin, Ireland. doi:10.3115/v1/S14-2001.

McCann, B., Bradbury, J., Xiong, C., & Socher, R. (2017). Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems* (pp. 6294–6305). Long Beach, California, USA. URL: http://papers.nips.cc/paper/7209-learned-in-translation-contextualized-word-vectors.pdf.

Mou, L., Men, R., Li, G., Xu, Y., Zhang, L., Yan, R., & Jin, Z. (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)* (pp. 130–136). Berlin, Germany. doi:10.18653/v1/P16-2022.

Pang, B., & Lee, L. (2004). A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd annual meeting on Association for Computational Linguistics* (p. 271). Barcelona, Spain. doi:10.3115/1218955.1218990.

Pang, B., & Lee, L. (2005). Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd annual meeting on association for computational linguistics* (pp. 115–124). Ann Arbor, Michigan. doi:10.3115/1219840.1219855.

Pennington, J., Socher, R., & Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (pp. 1532–1543). Doha, Qatar. doi:10.3115/v1/D14-1162.

Qian, Q., Tian, B., Huang, M., Liu, Y., Zhu, X., & Zhu, X. (2015). Learning tag embeddings and tag-specific composition functions in recursive neural network. 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)* (pp. 1365–1374). Beijing, China. doi:10.3115/v1/P15-1132.

Radford, A., Jozefowicz, R., & Sutskever, I. (2017). Learning to generate reviews and discovering sentiment. *arXiv preprint arXiv:1704.01444* , . URL: https://arxiv.org/pdf/1704.01444.pdf.

Shen, G., Deng, Z.-h., Huang, T., & Chen, X. (2020). Learning to compose over tree structures via pos tags for sentence representation. *Expert Systems with Applications*, *141*, 112917. doi:https://doi.org/10.1016/j.eswa.2019.112917.

Shen, T., Zhou, T., Long, G., Jiang, J., Pan, S., & Zhang, C. (2018). Disan: Directional self-attention network for rnn/cnn-free language understanding. In *Thirty-Second AAAI Conference on Artificial Intelligence* (pp. 5446–5455). New Orleans, USA. URL: https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16126/16099.

Socher, R., Bauer, J., Manning, C. D., & Ng, A. Y. (2013a). Parsing with compositional vector grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 455–465). Sofia, Bulgaria. URL: https://www.aclweb.org/anthology/P13-1045.pdf.

Socher, R., Huval, B., Manning, C. D., & Ng, A. Y. (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* (pp. 1201–1211). Jeju Island, Korea. URL: http://dl.acm.org/citation.cfm?id=2390948.2391084.

Socher, R., Lin, C. C., Manning, C., & Ng, A. Y. (2011). Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th international conference on machine learning* (pp. 129–136). Bellevue, Washington, USA. URL: https://nlp.stanford.edu/pubs/SocherLinNgManning_ICML2011.pdf.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., & Potts, C. (2013b). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing* (pp. 1631–1642). Seattle, Washington, USA. URL: https://www.aclweb.org/anthology/D13-1170.

Tai, K. S., Socher, R., & Manning, C. D. (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)* (pp. 1556–1566). Beijing, China. doi:10.3115/v1/P15-1150.

Teng, Z., & Zhang, Y. (2017). Head-lexicalized bidirectional tree lstms. *Transactions of the Association for Computational Linguistics*, *5*, 163–177. URL: https://doi.org/10.1162/tacl_a_00053.

Wang, Y., Li, S., Yang, J., Sun, X., & Wang, H. (2017a). Tag-enhanced tree-structured neural networks for implicit discourse relation classification. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)* (pp. 496–505). Taipei, Taiwan. URL: https://www.aclweb.org/anthology/I17-1050.

Wang, Z., Hamza, W., & Florian, R. (2017b). Bilateral multi-perspective matching for natural language sentences. In *Proceedings of the 26th International*

Joint Conference on Artificial Intelligence (pp. 4144–4150). Melbourne, Australia. URL: https://www.ijcai.org/proceedings/2017/0579.pdf.

Xu, C., Lin, Z., Wu, S., & Wang, H. (2019). Multi-level matching networks for text matching. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 949–952). Paris, France. doi:10.1145/3331184.3331276.

Yang, B., Wong, D. F., Xiao, T., Chao, L. S., & Zhu, J. (2017). Towards bidirectional hierarchical representations for attention-based neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing* (pp. 1432–1441). Copenhagen, Denmark. doi:10.18653/v1/D17-1150.

Zhang, Y., Zhang, Z., Miao, D., & Wang, J. (2019). Three-way enhanced convolutional neural networks for sentence-level sentiment classification. *Information Sciences*, *477*, 55–64. URL: https://doi.org/10.1016/j.ins.2018.10.030.

Zhou, P., Qi, Z., Zheng, S., Xu, J., Bao, H., & Xu, B. (2016). Text classification improved by integrating bidirectional lstm with two-dimensional max pooling. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers* (pp. 3485–3495). Osaka, Japan. URL: https://www.aclweb.org/anthology/C16-1329.

Zhou, Q., Wang, X., & Dong, X. (2018). Differentiated attentive representation learning for sentence classification. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence* (pp. 4630–4636). Stockholm, Sweden. URL: https://www.ijcai.org/proceedings/2018/0644.pdf.

Zhu, X., Sobihani, P., & Guo, H. (2015). Long short-term memory over recursive structures. In *Proceedings of the 32nd International Conference on Machine Learning* (pp. 1604–1612). Lille, France. URL: http://proceedings.mlr.press/v37/zhub15.pdf.