TreeLSTM with Tag-Aware Hypernetwork 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*

Abstract

Tree-structured neural networks, such as TreeLSTM and its variants, have proven effective for learning semantic representations of sentences, which are useful for a variety of tasks in natural language processing such as text categorisation, text semantic matching and machine translation. These neural network models take as inputs parse trees of sentences, which are generated by a language parser. However, most existing tree-structured neural network models lack the ability of distinguishing different syntactic compositions, thus the expressive power of these models is limited. Moreover, the syntactic knowledge provided by Part-of-Speech tags in a parse tree has not been fully utilised in existing tree-structured neural network models. It is expected that such syntactic knowledge should help distinguish syntactic compositions, so should result in better semantic representation. This paper proposes a novel neural network model, TagHyperTreeLSTM, which contains two components, a tag-aware hypernetwork and a sentence encoder. The tag-aware hypernetwork, which accepts tags as inputs, generates the parameters of the sentence encoder dynamically in order to distinguish different syntactic compositions. The sentence encoder, which accepts words as inputs, generates the final sentence representation. Experimental results show that the proposed model achieves superior or competitive performance in text classification and text semantic matching based on six

*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)

benchmark datasets when compared against previous tree-structured models.

Keywords: Sentence representation, Text classification, Text semantic matching, Dynamic composition, TreeLSTM

1. Introduction

Representing a sentence as a compact semantic vector is a fundamental operation for various natural language processing (NLP) tasks, such as text classification [1, 2], machine translation [3, 4], text semantic matching [5, 6]. Recently, deep neural networks such as recurrent neural network (RNN) [7] and its variants including Long Short-Term Memory (LSTM) [8] and Gated Recurrent Unit (GRU) [3] are widely used to represent sentence-level information by converting a sequence of words into a fixed length vector. In addition to RNNs, other types of neural networks such as convolutional neural networks (CNNs) [1] and self-attention based models [9] are also used for sentence representation. However, the above three types of models take a sentence as a flat sequence, without considering the structural information of the sentence.

In contrast, tree-structured neural networks or recursive neural networks (RecNNs) [10, 11] take a sentence as a recursive structure. Based on the pre-obtained syntactic parse tree, a RecNN model converts each word at a leaf node of the tree to a representation vector, and then uses a composition function to compose word/phrase pairs to get representations of the intermediate nodes of the tree. Finally, the representation of the root node is viewed as a representation of the sentence. However, a major limitation is that all kinds of compositions share the same parameters in a RecNN model as shown in Figure 1, neglecting the fact that different syntactic compositions exist which require different parameters for the RecNN model to handle precisely.

In order to distinguish different syntactic compositions, some dynamic compositional models are proposed, in which different composition functions are used for different compositions. One way of performing dynamic composition is

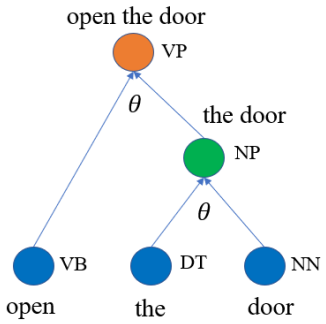


Figure 1: A RecNN model where the parameter θ is shared by different kinds of syntactic compositions such as *verb-noun* (VP) composition and *determiner-noun* (NP) composition.

with the aid of Part-of-Speech (POS) tags[1] attached to each node in the syntactic tree of a sentence [12, 13, 14, 15]. Usually, a low-dimensional distributed vector is used to represent each tag, namely tag embedding. Tag embeddings are learned and then used, together with the word embeddings, as inputs to the composition function of the model. For example, Huang *et al.* [13] proposed TE-LSTM which takes tag embeddings as additional inputs to the gate functions of the TreeLSTM. A limitation of this kind of model is that the learned tag embeddings in these models are too simple to reflect the rich information that tags provide in different syntactic structures.

Another way of conducting dynamic composition is to take advantage of hypernetworks [16] which use a small network (i.e."hypernetwork") to generate the weights for a larger network (i.e. main network). Liu *et al.* [2] proposed DC-TreeLSTM which is composed of two separate TreeLSTMs with similar structures but different numbers of parameters. The smaller TreeLSTM is employed to calculate the weights of the bigger TreeLSTM. A limitation of DC-TreeLSTM is that the two TreeLSTMs share the same inputs, i.e. word embeddings, thus, the model can only extract semantic information and lacks the ability of capturing syntactic information which is useful for dynamic composition. Although

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

Shen et al. [15] considered syntactic information and improved the performance of DC-TreeLSTM by incorporating tag information and word information in the hypernetwork TreeLSTM, tag information is only a supplement to the word information. However, comparing with words, tags can help the model to distinguish different syntactic compositions more explicitly. Thus, how to use tags in a more efficient way for dynamic composition still need to be further explored.

In short, the syntactic information of a sentence has not been fully explored for dynamic composition in previous studies. To alleviate this limitation, we propose a new model, TagHyperTreeLSTM, which is composed of a tag-aware hypernetwork and a sentence encoder. The purpose of the tag-aware hypernetwork is two-fold: (1) to extract much more syntactic information by encoding some structural information into tag embeddings; and (2), to dynamically generate parameters for the sentence encoder. Specifically, the tag-aware hypernetwork is a standard TreeLSTM which only accepts tag embeddings at each node of a tree, and outputs a new tag representation of the node. Then these new tag representations, which encode structural information of nodes, will be used to generate parameters for the sentence encoder to perform dynamic composition. The sentence encoder is another TreeLSTM which accepts words as inputs and outputs the final sentence representation. We evaluate the proposed TagHyperTreeLSTM model on two typical NLP tasks: text classification and text semantic matching. The results show that TagHyperTreeLSTM is more expressive than previous models due to its ability of capturing both semantic and syntactic information.

The contributions of the paper can be summarised as follows:

- We propose a new perspective on the usage of tags in a syntactic tree and devise a novel dynamic compositional model TagHyperTreeLSTM for sentence representation.
- Experimental results show that the proposed model achieves state-of-the-art performance among tree-structured models on six benchmark datasets with fewer parameters.

- An elaborate qualitative analysis is presented, giving an intuitive explanation of why our model works.

2. Related Works

Recursive neural networks (RecNNs) or tree-structured neural networks is one type of neural architecture which learns sentence representation by exploiting syntactic structures. Earlier researches on RecNNs mainly focus on investigating effective composition functions. Socher et al. [10] firstly proposed the RecNN architecture and uses a simple feed-forward neural network as the composition function of the model. Latterly, some more complex composition functions such as matrix-vector multiplication [17], tensor computation [11] and TreeLSTM [18] are proposed to improve the performance of the basic RecNN. Benefiting from considering the syntactic structure of sentences, RecNNs achieves impressive performance on a variety of NLP tasks. However, a major limitation of these RecNN models is that the same composition function is used recursively over the syntactic tree, thus lacking the ability of distinguishing different syntactic compositions.

To alleviate this problem, a straightforward method is to utilize multiple composition functions. Socher et al. [19] selected a suitable composition function for each word/phrase pair according to its syntactic categories. Similarly, Dong et al. [20] introduced AdaRNN, which also uses multiple composition functions and adaptively selects them depending on tags and child vectors. However, the predefined composition functions are usually designed manually for some specific tasks, thus the generalization ability of these models is limited. Recently, researchers proposed some models which can automatically perform dynamic composition with the aid of tags [12, 14, 13, 21]. Generally, tags are usually used as supplementary inputs for RecNNs in these models. For example, Wang et al. [14] and Huang et al. [13] employed tag embeddings as additional inputs to control the gates of TreeLSTM to conduct dynamic composition.

Some other works such as [2, 15] took advantage of recent works on dynam-

ic parameter prediction [16], and proposed to use a hypernetwork to generate parameters for RecNNs dynamically. Liu et al. [2] proposed DC-TreeLSTM which is composed of two separate TreeLSTMs with similar structure but different number of parameters. The smaller TreeLSTM is employed to calculate the weights of the bigger TreeLSTM. The two TreeLSTMs in a DC-TreeLSTM model share the same input, i.e. word embeddings. Shen et al. [15] proposed TG-HTreeLSTM, which improves the performance of DC-TreeLSTM by designing a complex information fusion layer which incorporates tag information and word information for hypernetwork TreeLSTM. These two works are very related to our work, but with significant difference. The hypernetwork in our model is solely based on tags without using any word information. Moreover, experimental results show that the proposed model outperforms TG-HTreeLSTM with fewer parameters.

3. The Model

In this section, we present a novel dynamic compositional neural architecture, named **TagHyperTreeLSTM**, which consists of two components, i.e. tag-aware hypernetwork and sentence encoder. The tag-aware hypernetwork is a standard TreeLSTM which only accepts tag embeddings as inputs and outputs new tag representation of each node in a parse tree. The sentence encoder accepts word representations as inputs and outputs sentence representations. The parameters of sentence encoder are calculated based on the outputs of tag-aware hypernetwork. Figure 2 illustrates the proposed model.

In subsection 3.1, the standard TreeLSTM architecture is outlined. Subsection 3.2 describes the tag-aware hypernetwork and subsection 3.3 presents the sentence encoder.

3.1. TreeLSTM

LSTM [8] is proposed to deal with the vanishing and exploding gradient problems of RNN [7], which can capture long-distance dependencies for sequential

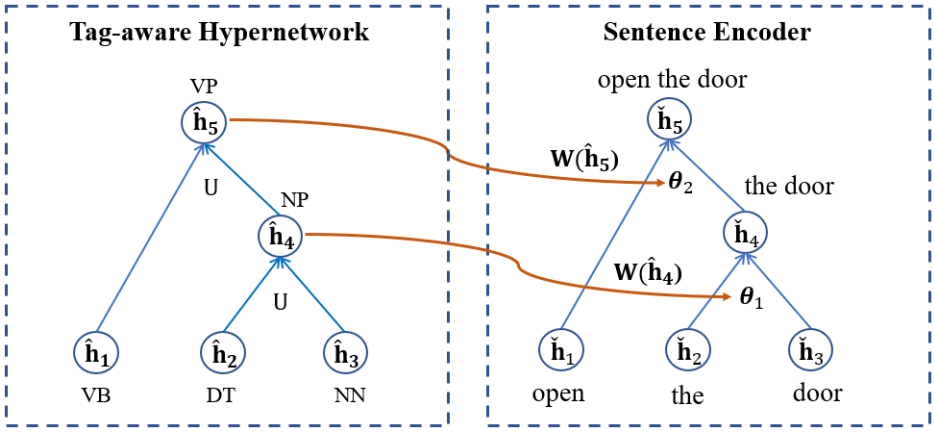


Figure 2: An overview of the TagHyperTreeLSTM. The tag-aware hypernetwork is a standard TreeLSTM which only accepts tag embeddings as inputs. $\mathbf{U}$ denotes parameters of tag-aware hypernetwork. $(\hat{\mathbf{h}}_1, \ldots, \hat{\mathbf{h}}_5)$ and $(\check{\mathbf{h}}_1, \ldots, \check{\mathbf{h}}_5)$ are hidden states of nodes in tag-aware hypernetwork and sentence encoder, respectively. $\mathbf{W}(\hat{\mathbf{h}}_4)$ and $\mathbf{W}(\hat{\mathbf{h}}_5)$ are intermediate hidden vectors computed based on $\hat{\mathbf{h}}_4$ and $\hat{\mathbf{h}}_5$. Parameters of the sentence encoder at each non-leaf node, i.e. θ_1 and θ_2, are not static, but are changed dynamically by the hidden vectors $\mathbf{W}(\hat{\mathbf{h}}_4)$ and $\mathbf{W}(\hat{\mathbf{h}}_5)$, respectively, and the detailed computation refers to subsection 3.3.

data due to its well-designed gate mechanism. TreeLSTM [18] is a generalization of LSTM to tree-structured network topologies which achieves impressive performance for sentence representation.

For each node j in a binary constituency tree of a sentence, let $\mathbf{x}_j = [x_1, \ldots, x_{d_e}]^T$ be an input vector, $\mathbf{h}_j^l = [h_1^l, \ldots, h_d^l]^T$ and $\mathbf{h}_j^r = [h_1^r, \ldots, h_d^r]^T$ be the hidden states of left child and right child, respectively. $\mathbf{c}_j^l = [c_1^l, \ldots, c_d^l]^T$ and $\mathbf{c}_j^r = [c_1^r, \ldots, c_d^r]^T$ be the memory cells of left child and right child, respectively. The composition function of a TreeLSTM unit can be described as follows:

$$\mathbf{i}_j = \sigma(\mathbf{W}_i[\mathbf{x}_j; \mathbf{h}_j^l; \mathbf{h}_j^c] + \mathbf{b}_i) \tag{1}$$

$$\mathbf{f}_j^l = \sigma(\mathbf{W}_l[\mathbf{x}_j; \mathbf{h}_j^l; \mathbf{h}_j^c] + \mathbf{b}_l) \tag{2}$$

$$\mathbf{f}_j^r = \sigma(\mathbf{W}_r[\mathbf{x}_j; \mathbf{h}_j^l; \mathbf{h}_j^c] + \mathbf{b}_r) \tag{3}$$

$$\mathbf{g}_j = \tanh(\mathbf{W}_g[\mathbf{x}_j; \mathbf{h}_j^l; \mathbf{h}_j^c] + \mathbf{b}_g) \tag{4}$$

$$\mathbf{o}_j = \sigma(\mathbf{W}_o[\mathbf{x}_j; \mathbf{h}_j^l; \mathbf{h}_j^c] + \mathbf{b}_o) \tag{5}$$

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

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

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

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).

$$[\mathbf{h}_j; \mathbf{c}_j] = \mathbf{TreeLSTM}(\mathbf{x}_j, \mathbf{h}_j^l, \mathbf{h}_j^r, \mathbf{c}_j^l, \mathbf{c}_j^r) \tag{8}$$

3.2. Tag-aware Hypernetwork

In this subsection, the tag-aware hypernetwork in Figure 2 is described in detail. It is a standard TreeLSTM but accepts only tag embeddings as inputs. The purpose of the tag-aware hypernetwork is to generate parameters dynamically for the sentence encoder (the right side of Figure 2).

Formally, we denote tag embedding for the tag attached to each node j in a binary constituency tree as $\mathbf{e}_j = [e_1, \ldots, e_{d_t}]^T$. Then the hidden state $\hat{\mathbf{h}}_j \in \mathbb{R}^{d_t}$ and memory cell $\hat{\mathbf{c}}_j \in \mathbb{R}^{d_t}$ of node j are defined in the following way. If node j is a leaf node:

$$[\hat{\mathbf{h}}_j; \hat{\mathbf{c}}_j] = \tanh(\mathbf{V}\mathbf{e}_j + \mathbf{a}) \tag{9}$$

If node j is a non-leaf node:

$$[\hat{\mathbf{h}}_j; \hat{\mathbf{c}}_j] = \mathbf{TreeLSTM}(\mathbf{e}_j, \hat{\mathbf{h}}_j^l, \hat{\mathbf{h}}_j^r, \hat{\mathbf{c}}_j^l, \hat{\mathbf{c}}_j^r) \tag{10}$$

where **TreeLSTM** refers to Equation (8). $\mathbf{V} \in \mathbb{R}^{2d_t \times d_t}$ and $\mathbf{a} \in \mathbb{R}^{2d_t}$ are trainable parameters. The remaining notation follows Equations (1-7).

3.3. Sentence Encoder

In this subsection, we introduce the sentence encoder (the right side of Figure 2) which is used to compose word/phrase pair recursively over a binary constituency tree.

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 of the sentence as $(\mathbf{v}_1, \dots \mathbf{v}_m)$ where $\mathbf{v}_i = [v_1, \dots, v_{d_w}]^T, i \in [1, m]$. Note that these words are leaf nodes in the constituency tree generated by a parser. Instead of using word embeddings directly, we firstly use a LSTM [8] on them, and then use each hidden state and memory cell of the LSTM as inputs for leaf nodes in the sentence encoder, which is effective for performance improvements on several NLP tasks [22].

Thus, for leaf node t, the hidden state $\bar{\mathbf{h}}_t \in \mathbb{R}^{d_h}$ and memory cell $\bar{\mathbf{c}}_t \in \mathbb{R}^{d_h}$ are computed in sequential order, with the corresponding input, i.e. word embedding $\mathbf{v}_t \in \mathbb{R}^{d_m}$ in the following way:

$$[\bar{\mathbf{h}}_t; \bar{\mathbf{c}}_t] = \mathbf{LSTM}(\bar{\mathbf{h}}_{t-1}, \bar{\mathbf{c}}_{t-1}, \mathbf{v}_t) \tag{11}$$

where $\bar{\mathbf{h}}_{t-1} \in \mathbb{R}^{d_h}$ and $\bar{\mathbf{c}}_{t-1} \in \mathbb{R}^{d_h}$ refer to the hidden state and memory cell of LSTM at $(t-1)^{th}$ time-step. The hidden state $\bar{\mathbf{h}}_t$ and memory cell $\bar{\mathbf{c}}_t$ can be utilized as inputs to the sentence representation TreeLSTM, with the left (right) child of the target node j corresponding to the t^{th} word in the input sentence as following:

$$[\check{\mathbf{h}}_j^{\{l,r\}}; \check{\mathbf{c}}_j^{\{l,r\}}] = [\bar{\mathbf{h}}_t; \bar{\mathbf{c}}_t] \tag{12}$$

Then, for each none-leaf node j, a TreeLSTM with dynamic parameters is

used to obtain its hidden state $\check{\mathbf{h}}_j \in \mathbb{R}^{d_h}$ and memory cell $\check{\mathbf{c}}_j \in \mathbb{R}^{d_h}$ as follows:

$$\mathbf{W}(\hat{\mathbf{h}}_j) = \mathbf{W}_d \hat{\mathbf{h}}_j + \mathbf{b}_d \tag{13}$$

$$\check{\mathbf{i}}_j = \sigma(\mathbf{W}(\hat{\mathbf{h}}_j) \odot \mathbf{W}_i[\check{\mathbf{h}}_j^l; \check{\mathbf{h}}_j^c] + \mathbf{W}(\hat{\mathbf{h}}_j) \odot \check{\mathbf{b}}_i) \tag{14}$$

$$\check{\mathbf{f}}_j^l = \sigma(\mathbf{W}(\hat{\mathbf{h}}_j) \odot \mathbf{W}_l[\check{\mathbf{h}}_j^l; \check{\mathbf{h}}_j^c] + \mathbf{W}(\hat{\mathbf{h}}_j) \odot \check{\mathbf{b}}_l) \tag{15}$$

$$\check{\mathbf{f}}_j^r = \sigma(\mathbf{W}(\hat{\mathbf{h}}_j) \odot \mathbf{W}_r[\check{\mathbf{h}}_j^l; \check{\mathbf{h}}_j^c] + \mathbf{W}(\hat{\mathbf{h}}_j) \odot \check{\mathbf{b}}_r) \tag{16}$$

$$\check{\mathbf{g}}_j = \sigma(\mathbf{W}(\hat{\mathbf{h}}_j) \odot \mathbf{W}_g[\check{\mathbf{h}}_j^l; \check{\mathbf{h}}_j^c] + \mathbf{W}(\hat{\mathbf{h}}_j) \odot \check{\mathbf{b}}_g) \tag{17}$$

$$\check{\mathbf{o}}_j = \sigma(\mathbf{W}(\hat{\mathbf{h}}_j) \odot \mathbf{W}_o[\check{\mathbf{h}}_j^l; \check{\mathbf{h}}_j^c] + \mathbf{W}(\hat{\mathbf{h}}_j) \odot \check{\mathbf{b}}_o) \tag{18}$$

$$\check{\mathbf{c}}_j = \check{\mathbf{f}}_j^l \odot \check{\mathbf{c}}_j^l + \check{\mathbf{f}}_j^r \odot \check{\mathbf{c}}_j^r + \check{\mathbf{i}}_j \odot \check{\mathbf{g}}_j \tag{19}$$

$$\check{\mathbf{h}}_j = \check{\mathbf{o}}_j \odot \tanh(\check{\mathbf{c}}_j) \tag{20}$$

where $\mathbf{W}_d \in \mathbb{R}^{d_h \times d_h}$ and $\mathbf{b}_d \in \mathbb{R}^{d_h}$ are trainable parameters. $\mathbf{W}(\hat{\mathbf{h}}_j) \in \mathbb{R}^{d_h}$ is an intermediate hidden vector computed based on the output of the tag-aware hypernetwork at node j, which modifies the corresponding static parameters $\mathbf{W}_i, \mathbf{W}_l, \mathbf{W}_r, \mathbf{W}_g, \mathbf{W}_o \in \mathbb{R}^{d_h \times 2d_h}$ and $\check{\mathbf{b}}_i, \check{\mathbf{b}}_l, \check{\mathbf{b}}_r, \check{\mathbf{b}}_g, \check{\mathbf{b}}_o \in \mathbb{R}^{d_h}$ by scaling each row of weight matrix linearly by an element in vector. The remaining notation follows Equations (1-7). Finally, the hidden state of the root node $\check{\mathbf{h}}^{\mathbf{root}} \in \mathbb{R}^{d_h}$ is used as the representation for the given sentence.

4. Applications of TagHyperTreeLSTM

In this section, we describe the application of TagHyperTreeLSTM 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 single layer MLP with the ReLU activation function, followed by a softmax classifier to obtain the final predicted probability distribution of class y given sentence s as following:

$$\mathbf{h}_s = \mathbf{Relu}(\mathbf{W}_s \check{\mathbf{h}}^{\mathbf{root}} + \mathbf{b}_s) \tag{21}$$

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

where $\mathbf{h}_s \in \mathbb{R}^{d_s}$ is the intermediate feature vector for the softmax classifier. $\mathbf{W}_s \in \mathbb{R}^{d_s \times d_h}, \mathbf{b_s} \in \mathbb{R}^{d_s}, \mathbf{W}_c \in \mathbb{R}^{d_c \times d_s}, \mathbf{b_c} \in \mathbb{R}^{d_c}$ are trainable parameters.

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 TagHyperTreeLSTM is used to encode $s1$ and $s2$ into two sentence representation vectors $\check{\mathbf{h}}_{\mathbf{s1}}^{\mathbf{root}}, \check{\mathbf{h}}_{\mathbf{s2}}^{\mathbf{root}} \in \mathbb{R}^{d_h}$. Next, some matching heuristics [23] are used to combine the two sentence vectors together in the following way:

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

Then, a single layer MLP with the 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 labels.

5. Experiments

5.1. Datasets

We evaluate the proposed model on four benchmark datasets for text classification (SST2, MR, SUBJ, TREC) and two datasets (SICK and SNLI) for text semantic matching:

- SST2: Stanford Sentiment Treebank consisting of movie reviews with positive or negative label [11].[2]
- MR: The movie reviews with positive or negative label [24].[3]
- SUBJ: Sentences grouped as being either subjective or objective [25]. [4]
- TREC: A dataset which groups questions into six different question types [26].[5]
- SICK: A textual entailment dataset with three classes (entailment, neutral, contradiction) [27].[6]
- SNLI: The stanford natural language inference dataset with three classes (entailment, neutral, contradiction) [28].[7]

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

5.2. Experimental Setup

For all the datasets, sentences are tokenized and parsed by Stanford PCFG parser[8] [29]. Word embeddings are initialized with the 300-dimensional GloVe word vectors [30], and embeddings of out-of-vocabulary words and tags are initialized by randomly sampling from the uniform distribution $[-0.005, 0.005]$.

[2] https://nlp.stanford.edu/sentiment/
[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/projects/snli/
[8] https://nlp.stanford.edu/software/lex-parser.shtml

Dataset	Train	Dev	Test	$\mathbf{L_{avg}}$	|V|	|T|	Class
SST2	6920	872	1821	18	15K	73	2
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
SNLI	549K	9800	9800	10	36K	72	3

Table 1: Statistics of six benchmark datasets for two tasks. Train, Dev and Test are the size of training, 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.

Tag embeddings are fine-tuned during training procedure while word embeddings are fixed. Hidden size for tag-aware hypernetwork and the dimension of tag embeddings are fine-tuned from $[25, 50]$. Hidden size for sentence representation TreeLSTM is 100 for all datasets. Batch size is selected from $[32, 64]$. Weights of the model are trained by minimizing the cross-entropy of the training dataset by the Adadelta [31] optimizer. The initial learning rate is 1. The accuracy metric is used in this paper to measure the performance of the proposed and all comparison models on six datasets.

5.3. Results

Text Classification. The proposed model is compared with two kinds of models, i.e. tree-structured models and other neural models. Table 2 shows test accuracies of the proposed TagHyperTreeLSTM and all comparison models on four text classification datasets. Generally, compared with all baseline models, the proposed TagHyperTreeLSTM achieves superior or competitive performance on all four text classification datasets.

From table 2, we can observe that, firstly, compared with previous tree-structured models, TagHyperTreeLSTM sets a new state of the art on all four datasets - SST2, MR, SUBJ and TREC with accuracies of 91.2%, 83.7%, 95.2% and 96.0%, respectively.

Model	SST2	MR	SUBJ	TREC
Tree-structured neural models				
RecNN [10]	82.4	76.4	91.8	90.2
RNTN [11]	86.4	-	-	-
AdaMC-RNTN [20]	87.1	-	-	-
TE-RNN [12]	86.5	77.9	-	-
TreeLSTM [18]	88.0	-	-	-
AdaHT-LSTM [32]	87.8	81.9	94.1	-
TE-LSTM [13]	89.6	82.2	-	-
BiTreeLSTM [33]	90.3			94.8
DC-TreeLSTM [2]	87.8	81.7	93.7	93.8
Gumbel Tree-LSTM [22]	90.7	-	-	-
TG-HTreeLSTM [15]	90.4	82.6	94.9	95.8
Other neural models				
CNN [1]	88.1	81.5	93.4	93.6
LSTM [18]	84.9	-	-	-
BCN + Char + CoVe* [34]	90.3	-	-	95.8
byte-mLSTM* [35]	91.8	86.9	94.6	
DiSAN [9]	-	-	94.2	94.2
DARLM [36]	-	83.2	94.1	96.0
WSAN [37]	-	83.2	94.6	95.0
Transformer† [38]	86.9	80.2	94.1	91.9
star-Transformer† [39]	87.1	80.7	93.6	93.0
Bert*† [40]	90.8	85.8	96.0	96.8
TagHyperTreeLSTM (proposed)	**91.2**	**83.7**	**95.2**	**96.0**

Table 2: Accuracies of previous neural models and proposed model on four different text classification datasets. The symbol * indicates models that are pre-trained with large external corpora. The symbol † indicates our implementations.

Secondly, compared with DC-TreeLSTM and TG-HTreeLSTM which are two models very related to our work, TagHyperTreeLSTM outperforms them on all four datasets. DC-TreeLSTM is the first work that employs a hypernetwork to generate parameters dynamically for a TreeLSTM which is responsible for sentence representation. Note that the hypernetwork and the sentence representation TreeLSTM share the same input (i.e. word embeddings) in DC-TreeLSTM. Different from DC-TreeLSTM, the hypernetwork in our model is based on tag embeddings without using any word information. TagHyperTreeLSTM is superior to the DC-TreeLSTM on four datasets SST2, MR, SUBJ and TREC with 3.4%, 2%, 1.5% and 2.2% improvements, respectively. A possible reason is that tag information is more useful for distinguishing different syntactic compositions than word information. Moreover, TagHyperTreeLSTM is slightly better than the state-of-the-art model TG-HTreeLSTM on all four datasets SST2, MR, SUBJ and TREC with 0.8%, 1.1%, 0.3% and 0.2% improvements, respectively. TG-HTreeLSTM uses tag information as a supplement to word information and devises a complex information fusion layer for the hypernetwork. Compared with TG-HTreeLSTM, the proposed model achieves better performance with fewer parameters thus is more effective and efficient.

Thirdly, compared with other neural models, TagHyperTreeLSTM shows its consistently better performance on four datasets. Althogh Bert achieves better performance than TagHyperTreeLSTM, it is pre-trained with large external corpora while TagHyperTreeLSTM do not perform any pre-training. As table 2 shows, TagHyperTreeLSTM outperforms all the models that do not perform pre-training, including recently published transformer based model such as star-Transformer.

Text Semantic Matching. To evaluate the proposed TagHyperTreeLSTM on other NLP tasks, we also conduct an experiment on text semantic matching. Different from text classification requiring only one sentence at a time, each example in a text semantic matching dataset consists of two sentences. In this paper, we evaluate the performance of TagHyperTreeLSTM on SICK and SNLI datasets.

Model	Acc (%)
RNN [28]	72.2
LSTM [28]	77.6
RecNN[41]	74.9
RNTN [41]	76.9
MV-RNN [2]	75.5
DC-TreeLSTM [2]	82.3
TG-HTreeLSTM [15]	83.3
TagHyperTreeLSTM	83.9

Table 3: Accuracies of previous neural models and proposed model on the SICK dataset.

Model	Acc (%)
LSTM [28]	80.6
Tree-based CNN [23]	82.1
SPINN-PI [42]	83.2
Gumbel Tree-LSTM [22]	85.6
TagHyperTreeLSTM	85.5

Table 4: Accuracies of previous neural models and proposed model on the SNLI dataset.

Table 3 shows experimental results on SICK dataset. The performance of RNN, LSTM, RecNN, and RNTN are reported in [28]. The performance of MV-RNN is reported in [2], and performance of the other models come from respective papers. We can find that TagHyperTreeLSTM again demonstrates its superior performance compared against baseline models with an accuracy of 83.9%, and is slightly better than the best baseline model TG-HTreeLSTM with an improvement of 0.6%. Table 4 gives experimental results on SNLI dataset. For fair comparison, we only consider sentence-encoding based models. The performance of TagHyperTreeLSTM is on par with the previous tree-structured models. This comparison shows that TagHyperTreeLSTM is also effective in text sematic matching task, and has generalization ability in different NLP tasks.

Model	# Params	Acc (%)
CNN [1]	$360K$	88.1
LSTM [18]	$316K$	84.9
TreeLSTM [18]	$317K$	88.0
TE-LSTM [13]	$919K$	89.6
Gumbel Tree-LSTM [22]	$1M$	90.7
TG-HTreeLSTM [15]	$486K$	90.4
TagHyperTreeLSTM	**418K**	**91.2**

Table 5: Comparison of number of parameters of different models on SST2 dataset.

6. Analysis

6.1. An observation on model complexity

In this subsection, a comparison of the number of parameters between the proposed TagHyperTreeLSTM and some typical neural models on the SST2 dataset is presented in Table 5. Firstly, compared with basic TreeLSTM, the performance of TagHyperTreeLSTM is improved by 3.2%, but the number of parameters are only increased by about $100K$. Secondly, compared with previous models which also use tag information, such as TE-LSTM and TG-HTreeLSTM, the proposed TagHyperTreeLSTM outperforms them with fewer parameters. This comparison demonstrates that the proposed model is more effective and efficient than previous models.

6.2. Configuration Study

In this section, we present a configuration study on key modules of the proposed model to explore their effectiveness. As shown in Table 6, if we use word information instead of tag information as the input of the hypernetwork (the left side of Figure 2), the accuracy of the model drops to 95.6% on TREC dataset and 83.3% on SICK dataset, respectively. A possible reason for this performance degradation is that tag information is more explicit and useful in distinguishing different syntactic structures than word information. Moreover, we do not get much improvement (no improvement on TREC dataset and only

Model	TREC	SICK
TagHyperTreeLSTM	96.0	83.9
word	95.6	83.3
word + tag	96.0	84.0
-LSTM	95.2	83.3

Table 6: A configuration study on key modules of TagHyperTreeLSTM. Test accuracies on TREC and SICK datasets are reported. word: only use word information in hypernetwork. word + tag: use both word and tag information in hypernetwork. -LSTM: remove LSTM layer in sentence encoder.

0.1% improvement on SICK dataset) using both word and tag information in the hypernetwork. This may be because syntactic tags have provided enough information in distinguishing different syntactic structures, thus the contribution of word information is very limited. If we remove LSTM layer in sentence encoder (subsection 3.3), the accuracy of the model drops on both datasets with more than 0.5% degradation. Thus, using LSTM to get contextualized word representations as the inputs of the sentence encoder is crucial for performance improvement.

6.3. Qualitative Analysis

As described in previous sections, in order to distinguish different syntactic compositions, parameters of the sentence encoder at each non-leaf node j are not static, but are changed dynamically by a latent vector $\mathbf{W}(\hat{\mathbf{h}}_j)$, which is computed based on the hidden state of the tag-aware hypernetwork of node j. To get an intuitive understanding on how the latent vector $\mathbf{W}(\hat{\mathbf{h}}_j)$ works, we design an experiment to explore behaviours of neurons in $\mathbf{W}(\hat{\mathbf{h}}_j)$.

We randomly sample some sentences on the test set from the TREC dataset. Table 7 presents some interpretable neurons and some representative phrases or clauses captured by these neurons. By analyzing the maximum activation neurons in $\mathbf{W}(\hat{\mathbf{h}}_j)$ at each non-leaf node j, we find different neurons focus on capturing different syntactic compositions. For example, the $65th$, $15th$ and $30th$ neurons are more sensitive to *noun phrases* (NP) compositions, while the $88th$

	Node tags	Neurons	Child nodes tags	Examples
Phrase Level	WHADJP	$39th$	WRB+JJ	how+fast, how+much how+tall, how+far
	WHNP	$77th$	WDT+NN	what+continent, what+year what+color, what+province
	PP	$88th$	IN+NP	in+1913, of+yugoslavia for+june, in+algeria
		$95th$	IN+NP	in+a galon, in+a ton, in+a mile in+the neuschwanstein castle
	NP	$65th$	NNP+NNP	euphrates+river, eiffel+tower national+forest
		$15th$	JJ+NN	acid+rain, compounded+interest nuclear+power, trivial+pursuit
		$30th$	DT+NN	the+calculator, a+thyroid a+carcinogen, an+earthquake
			NN+NN	fuel+cell, state+flower yak+milk, spirometer+test
Clause Level	SQ	$91st$	VBZ+NP	is+acid rain is+john wayne airport
			VBP+NP	are+the rocky mountains are+in the troposphere
			VBD+NP	founded+american red cross invented+the hula hoop
		$37th$	VBZ+NP	is+the population of seattle is+the capital of mongolia

Table 7: Some interpretable neurons and the phrases/clauses captured by these neurons. Symbol + splits the left child and right child of current node.

and $95th$ neurons are more sensitive to *prepositional phrases* (PP) compositions. Moreover, note that although the $65th$, $15th$ and $30th$ neurons are sensitive to the same phrase type i.e. NP, the child node tags of the phrase are quite different. For example, the $65th$ neuron is sensitive to noun phrases composed of two *proper nouns* (NNP), while the $15th$ is sensitive to noun phrases composed of an *adjective* (JJ) and a *noun* (NN).

Moreover, we also find the occurrence of large value neurons in $\mathbf{W}(\hat{\mathbf{h}}_j)$ is dominated by nodes with specific syntactic structure or semantic basis which are significant for text classification and text semantic matching. Figures 3 - 5 show three examples. In Figure 3, the $9th$ neuron of $\mathbf{W}(\hat{\mathbf{h}}_j)$ is sensitive to sentences starting with "Who", which is crucial for the model to classify these sentences to the label "HUM". In Figure 4, the $10th$ neuron is sensitive to emotional terms, which can be regard as a sentinel, telling the sentence encoder network that an informative phrase is coming. We can see that the neuron has realized that the adjective phrase "quite fresh and delightful" is important for the final sentiment classification.

Figure 5 gives an example for text semantic matching. The $73rd$ neuron of $\mathbf{W}(\hat{\mathbf{h}}_j)$ monitors verb phrases such as "running down" and "standing still" which are more helpful for judging the semantic relation between the sentence pair "A man is standing still" and "A man is running down the road".

7. Conclusion

In this paper, we have presented a novel dynamic compositional architecture, named TagHyperTreeLSTM, for learning sentence representation, which has better expressive power due to its ability of distinguishing different syntactic compositions. The model consists of two components, i.e. sentence encoder and tag-aware hypernetwork. The purpose of the tag-aware hypernetwork is to extract syntactic information and to generate parameters dynamically for the sentence encoder. The sentence encoder is to output the final sentence representation. Experimental results on six datasets in two NLP tasks have

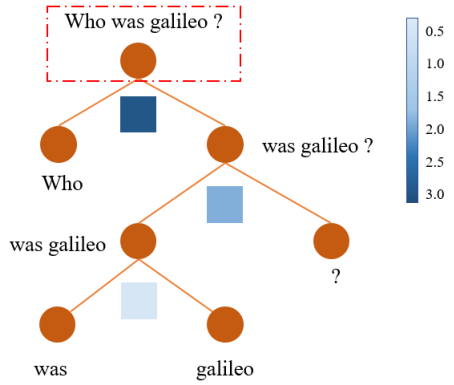


Figure 3: The visualization of parser tree and values of the $9th$ neuron of $\mathbf{W}(\hat{\mathbf{h}}_j)$ for sentence "Who was galileo ?". The color in the square below each non-leaf node j represents value of the $9th$ neuron of $\mathbf{W}(\hat{\mathbf{h}}_j)$.

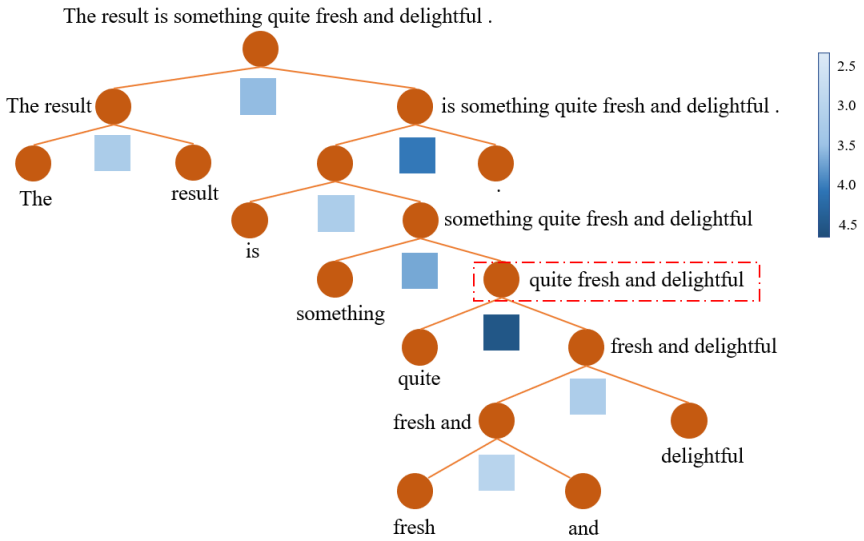
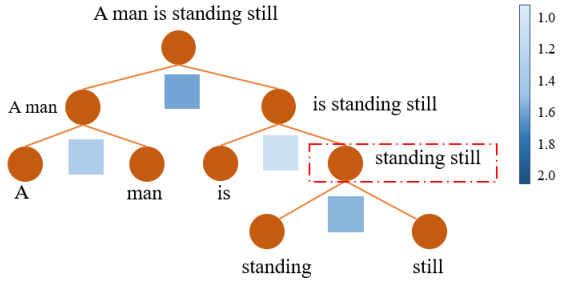
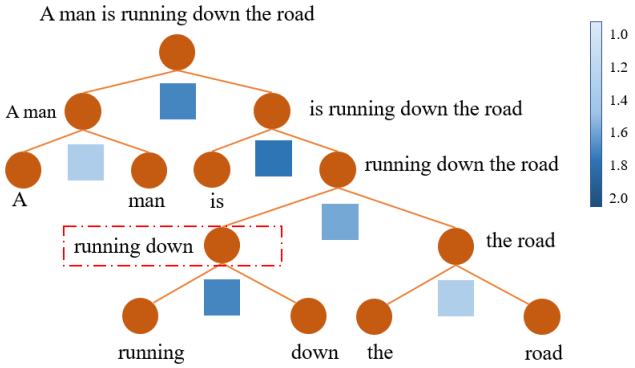


Figure 4: The visualization of parser tree and values of the $10th$ neuron of $\mathbf{W}(\hat{\mathbf{h}}_j)$ for sentence "The result is something quite fresh and delightful .". The color in the square below each non-leaf node j represents value of the $10th$ neuron of $\mathbf{W}(\hat{\mathbf{h}}_j)$.



(a) The visualization of parser tree and values of the $73rd$ neuron of $\mathbf{W}(\hat{\mathbf{h}}_j)$ for sentence "A man is standing still ".



(b) The visualization of parser tree and values of the $73rd$ neuron of $\mathbf{W}(\hat{\mathbf{h}}_j)$ for sentence "A man is running down the road ".

Figure 5: The visualization of parser tree and values of the $73rd$ neuron of $\mathbf{W}(\hat{\mathbf{h}}_j)$ for sentence pair "A man is running down the road/ A man is standing still". The color in the square below each non-leaf node j represents value of the $73rd$ neuron of $\mathbf{W}(\hat{\mathbf{h}}_j)$.

demonstrated the superiority and the generalization ability of the proposed model. Moreover, the qualitative analysis explains how our model works. In future work, we plan to extend the proposed TagHyperTreeLSTM to address other NLP tasks, such as question answering and machine translation.

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

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

[2] P. Liu, X. Qiu, X. Huang, Dynamic compositional neural networks over tree structure, in: Proceedings of the 26th International Joint Conference on Artificial Intelligence, Melbourne, Australia, 2017, pp. 4054–4060.

[3] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, 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), Doha, Qatar, 2014, pp. 1724–1734.

[4] J. Gehring, M. Auli, D. Grangier, Y. Dauphin, A convolutional encoder model for neural machine translation, in: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2017, pp. 123–135. doi:10.18653/v1/P17-1012.

[5] A. Conneau, D. Kiela, H. Schwenk, L. Barrault, A. Bordes, Supervised learning of universal sentence representations from natural language inference data, in: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, 2017, pp. 670–680. doi:10.18653/v1/D17-1070.

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

[7] J. L. Elman, Finding structure in time, Cognitive science 14 (2) (1990) 179–211. doi:https://doi.org/10.1207/s15516709cog1402_1.

[8] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural computation 9 (8) (1997) 1735–1780. doi:https://doi.org/10.1162/neco.1997.9.8.1735.

[9] T. Shen, T. Zhou, G. Long, J. Jiang, S. Pan, C. Zhang, Disan: Directional self-attention network for rnn/cnn-free language understanding, in: Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, USA, 2018, pp. 5446–5455.

[10] R. Socher, C. C. Lin, C. Manning, A. Y. Ng, Parsing natural scenes and natural language with recursive neural networks, in: Proceedings of the 28th international conference on machine learning, Bellevue, Washington, USA, 2011, pp. 129–136.

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

[12] Q. Qian, B. Tian, M. Huang, Y. Liu, X. Zhu, X. Zhu, 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), Beijing, China, 2015, pp. 1365–1374. doi:10.3115/v1/P15-1132.

[13] M. Huang, Q. Qian, X. Zhu, Encoding syntactic knowledge in neural networks for sentiment classification, ACM Transactions on Information Systems (TOIS) 35 (3) (2017) 26. doi:10.1145/3052770.

[14] Y. Wang, S. Li, J. Yang, X. Sun, H. Wang, 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), Taipei, Taiwan, 2017, pp. 496–505. doi:https://www.aclweb.org/anthology/I17-1050.

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

[16] D. Ha, A. Dai, Q. V. Le, Hypernetworks, arXiv preprint arXiv:1609.09106.

[17] R. Socher, B. Huval, C. D. Manning, A. Y. Ng, 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, Jeju Island, Korea, 2012, pp. 1201–1211.

[18] K. S. Tai, R. Socher, C. D. Manning, 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), Beijing, China, 2015, pp. 1556–1566. doi:10.3115/v1/P15-1150.

[19] R. Socher, J. Bauer, C. D. Manning, A. Y. Ng, Parsing with compositional vector grammars, in: Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Sofia, Bulgaria, 2013, pp. 455–465.

[20] L. Dong, F. Wei, C. Tan, D. Tang, M. Zhou, K. Xu, 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), Baltimore, Maryland, 2014, pp. 49–54. doi:10.3115/v1/P14-2009.

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

[22] J. Choi, K. M. Yoo, S.-g. Lee, Learning to compose task-specific tree structures, in: Thirty-Second AAAI Conference on Artificial Intelligence, 2018, pp. 5094–5101.

[23] L. Mou, R. Men, G. Li, Y. Xu, L. Zhang, R. Yan, Z. Jin, Natural language inference by tree-based convolution and heuristic matching, in: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, 2016, pp. 130–136.

[24] B. Pang, L. Lee, 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, Ann Arbor, Michigan, 2005, pp. 115–124. doi:10.3115/1219840.1219855.

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

[26] X. Li, D. Roth, Learning question classifiers, in: Proceedings of the 19th international conference on Computational linguistics-Volume 1, Taipei, Taiwan, 2002, pp. 1–7. doi:10.3115/1072228.1072378.

[27] M. Marelli, L. Bentivogli, M. Baroni, R. Bernardi, S. Menini, R. Zamparelli, 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), Dublin, Ireland, 2014, pp. 1–8. doi:10.3115/v1/S14-2001.

[28] S. R. Bowman, G. Angeli, C. Potts, C. D. Manning, A large annotated corpus for learning natural language inference, in: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, 2015, pp. 632–642. doi:10.18653/v1/D15-1075.

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

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

[31] M. D. Zeiler, Adadelta: an adaptive learning rate method, arXiv preprint arXiv:1212.5701.

[32] P. Liu, X. Qiu, X. Huang, Adaptive semantic compositionality for sentence modelling., in: Proceedings of the 26th International Joint Conference on Artificial Intelligence, Melbourne, Australia, 2017, pp. 4061–4067.

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

[34] B. McCann, J. Bradbury, C. Xiong, R. Socher, Learned in translation: Contextualized word vectors, in: Advances in Neural Information Processing Systems, 2017, pp. 6294–6305.

[35] A. Radford, R. Jozefowicz, I. Sutskever, Learning to generate reviews and discovering sentiment, arXiv preprint arXiv:1704.01444.

[36] Q. Zhou, X. Wang, X. Dong, Differentiated attentive representation learning for sentence classification, in: Proceedings of the 27th International Joint Conference on Artificial Intelligence, Stockholm, Sweden, 2018, pp. 4630–4636.

[37] T. Huang, Z. Deng, G. Shen, X. Chen, A window-based self-attention approach for sentence encoding, Neurocomputing 375 (2020) 25–31. doi:https://doi.org/10.1016/j.neucom.2019.09.024.

[38] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, in: Advances in neural information processing systems, 2017, pp. 5998–6008.

[39] Q. Guo, X. Qiu, P. Liu, Y. Shao, X. Xue, Z. Zhang, Star-transformer, in: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2019, pp. 1315–1325.

[40] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, in: Proceedings of

the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2019, pp. 4171–4186.

[41] S. R. Bowman, C. Potts, C. D. Manning, Recursive neural networks can learn logical semantics, in: Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality, 2015, pp. 12–21. doi:10.18653/v1/W15-4002.

[42] S. Bowman, J. Gauthier, A. Rastogi, R. Gupta, C. D. Manning, C. Potts, A fast unified model for parsing and sentence understanding, in: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, 2016, pp. 1466–1477.