Multiplicative Tree-Structured Long Short-Term Memory Networks for Semantic Representations

Nam Khanh Tran[*]
L3S Research Center
ntran@l3s.de

Weiwei Cheng
Amazon
weiweic@amazon.com

Abstract

Tree-structured LSTMs have shown advantages in learning semantic representations by exploiting syntactic information. Most existing methods model tree structures by bottom-up combinations of constituent nodes using the same shared compositional function and often making use of input word information only. The inability to capture the richness of compositionality makes these models lack expressive power. In this paper, we propose *multiplicative tree-structured LSTMs* to tackle this problem. Our model makes use of not only word information but also relation information between words. It is more expressive, as different combination functions can be used for each child node. In addition to syntactic trees, we also investigate the use of Abstract Meaning Representation in tree-structured models, in order to incorporate both syntactic and semantic information from the sentence. Experimental results on common NLP tasks show the proposed models lead to better sentence representation and AMR brings benefits in complex tasks.

1 Introduction

Learning the distributed representation for long spans of text from its constituents has been a crucial step of various NLP tasks such as text classification (Zhao et al., 2015; Kim, 2014), semantic matching (Liu et al., 2016), and machine translation (Cho et al., 2014). Seminal work uses recurrent neural networks (RNN) (Elman, 1990), convolutional neural networks (Kalchbrenner et al., 2014), and tree-structured neural networks (Socher et al., 2011; Tai et al., 2015) for sequence and tree modeling. Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) networks are a type of recurrent neural network that are capable of learning long-term dependencies across sequences and have achieved significant improvements in a variety of sequence tasks. LSTM has been extended to model tree structures (e.g., TreeLSTM) and produced promising results in tasks such as sentiment classification (Tai et al., 2015; Zhu et al., 2015) and relation extraction (Miwa and Bansal, 2016).

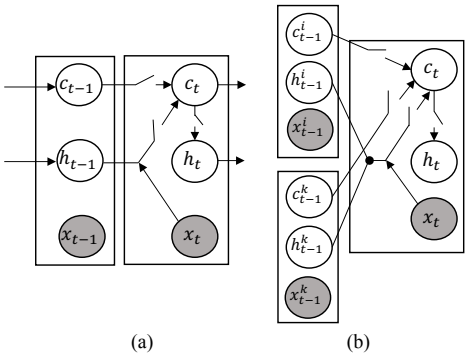

Figure 1: Topology of sequential LSTM and TreeLSTM: (a) nodes in sequential LSTM and (b) nodes in tree-structured LSTM

Figure 1 shows the topologies of the conventional chain-structured LSTM (Hochreiter and Schmidhuber, 1997) and the TreeLSTM (Tai et al., 2015), illustrating the input (x), cell (c) and hidden node (h) at a time step t. The key difference between Figure 1 (a) and (b) is the branching factor. While a cell in the sequential LSTM only depends on the single previous hidden node, a cell in the tree-structured LSTM depends on the hidden states of child nodes.

Despite their success, the tree-structured models have a limitation in their inability to fully capture the richness of compositionality (Socher et al., 2013a). The same combination function is used for all kinds of semantic compositions, though the

[*] Work done as an intern at Amazon.

*Proceedings of the 7th Joint Conference on Lexical and Computational Semantics (*SEM)*, pages 276–286
New Orleans, June 5-6, 2018.

compositions have different characteristics in nature. For example, the composition of the adjective and the noun differs significantly from the composition of the verb and the noun.

To alleviate this problem, some researchers propose to use multiple compositional functions, which are predefined according to some partition criterion (Socher et al., 2012, 2013a; Dong et al., 2014). Socher et al. (2013a) defined different compositional functions in terms of syntactic categories, and a suitable compositional function is selected based on the syntactic categories. Dong et al. (2014) introduced multiple compositional functions and a proper one is selected based on the input information. These models accomplished their objective to a certain extent but they still face critical challenges. The predefined compositional functions cannot cover all the compositional rules and they add much more learnable parameters, bearing the risk of overfitting.

In this paper, we propose *multiplicative TreeLSTM*, an extension to the TreeLSTM model, which injects relation information into every node in the tree. It allows the model to have different semantic composition matrices to combine child nodes. To reduce the model complexity and keep the number of parameters manageable, we define the composition matrices using the product of two dense matrices shared across relations, with an intermediate diagonal matrix that is relation dependent.

Though the syntactic-based models have shown to be promising for compositional semantics, they do not make full use of the linguistic information. For example, semantic nodes are often the argument of more than one predicate (e.g., coreference) and it is generally useful to exclude semantically vacuous words like articles or complementizers, i.e., leave nodes unattached that do not add further meaning to the resulting representations. Recently, Banarescu et al. (2013) introduced Abstract Meaning Representation (AMR), single rooted, directed, acyclic graphs that incorporate semantic roles, correference, negation, and other linguistic phenomena. In this paper, we investigate a combination of the semantic process provided by TreeLSTM model with the lexical semantic representation of the AMR formalism. This differs from most of existing work in this area, where syntactic rather than semantic information is incorporated to the tree-structured models. We seek to answer the question: *To what extent can we do better with AMR as opposed to syntactic representations, such as constituent and dependency trees, in tree-structured models?*

We evaluate the proposed models on three common tasks: sentiment classification, sentence relatedness, and natural language inference. The results show that the multiplicative TreeLSTM models outperform TreeLSTM models on the same tree structures. The results further suggest that using AMR as the backbone for tree-structured models is helpful in the complex task such as for longer sentences in natural language inference but not in sentiment classification, where lexical information alone suffices.

In short, our contribution is twofold:

1. We propose the new multiplicative TreeLSTM model that effectively learns distributed representation of a given sentence from its constituents, utilizing not only the lexical information of words, but also the relation information between the words.
2. We conduct an extensive investigation on the usefulness of lexical semantic representation induced by AMR formalism in tree-structured models.

2 Tree-Structured LSTM

A standard LSTM processes a sentence in a sequential order, e.g., from left to right. It estimates a sequence of hidden vectors given a sequence of input vectors, through the calculation of a sequence of hidden cell vectors using a gate mechanism. Extending the standard LSTM from linear chains to tree structures leads to TreeLSTM. Unlike the standard LSTM, TreeLSTM allows richer network topologies, where each LSTM unit is able to incorporate information from multiple child units.

As in standard LSTM units, each TreeLSTM unit contains input gate i_j, output gate o_j, a memory cell c_j, and hidden state h_j for node j. Unlike the standard LSTM, in TreeLSTM the gating vectors and the memory cell updates are dependent on the states of one or more child units. In addition, the TreeLSTM unit contains one forget gate f_{jk} for each child k instead of having a single forget gate. The transition equations of node j are as

follows:

$$\begin{aligned}
\tilde{h}_j &= \sum_{k \in C(j)} h_k \,, \\
i_j &= \sigma\left(W^{(i)} x_j + U^{(i)} \tilde{h}_j + b^{(i)}\right) \,, \\
o_j &= \sigma\left(W^{(o)} x_j + U^{(o)} \tilde{h}_j + b^{(o)}\right) \,, \\
f_{jk} &= \sigma\left(W^{(f)} x_j + U^{(f)} h_k + b^{(f)}\right) \,, \\
u_j &= \tanh\left(W^{(u)} x_j + U^{(u)} \tilde{h}_j + b^{(u)}\right) \,, \\
c_j &= i_j \odot u_j + \sum_{k \in C(j)} f_{jk} \odot c_k \,, \\
h_j &= o_j \odot \tanh(c_j) \,,
\end{aligned} \tag{1}$$

where $C(j)$ is the set of children of node j, $k \in C(j)$ in f_{jk}, σ is the sigmoid function, and $\odot$ is element-wise (Hadamard) product. $W^{(*)}$, $U^{(*)}$, $b^{(*)}$ are model parameters with $* \in \{u, o, i, f\}$.[1]

3 Multiplicative Tree-Structured LSTM

Encoding rich linguistic analysis introduces many distinct edge types or relations between nodes, such as syntactic dependencies and semantic roles. This opens up many possibilities for parametrization, but was not considered in most existing syntax-aware LSTM approaches, which only make use of input node information.

In this paper, we fill this gap by proposing multiplicative TreeLSTM, an extension to the TreeLSTM model, injecting relation information into every node in the tree. The multiplicative TreeLSTM model, *mTreeLSTM* for short, introduces more fined-grained parameters based on the edge types. Inspired by the multiplicative RNN (Sutskever et al., 2011), the hidden-to-hidden propagation in mTreeLSTM contains a separately learned transition matrix W_{hh} for each possible edge type and is given by

$$\tilde{h}_j = \sum_{k \in C(j)} W_{hh}^{r(j,k)} h_k \,, \tag{2}$$

where $r(j,k)$ signifies the connection type between node k and its parent node j. This parametrization is straightforward, but requires a large number of parameters when there are many edge types. For instance, there are dozens of syntactic edge types, each corresponding to a Stanford dependency label.

[1] In Tai et al. (2015), the TreeLSTM defined in Eq. (1) was referred to as *child-sum TreeLSTM*, which is a good choice for trees with high branching factor.

To reduce the number of parameters and leverage potential correlation among fine-grained edge types, we learned an embedding of the edge types and factorized the transition matrix $W_{hh}^{r(j,k)}$ by using the product of two dense matrices shared across edge types, with an intermediate diagonal matrix that is edge-type dependent:

$$W_{hh}^{r(j,k)} = W_{hm} \mathrm{diag}(W_{mr} e_{jk}) W_{mh} \,, \tag{3}$$

where e_{jk} is the edge-type embedding and is jointly trained with other parameters. The mapping from h_k to $\tilde{h}_j$ is then given by

$$\begin{aligned}
m_{jk} &= (W_{mr} e_{jk}) \odot (W_{mh} h_k) \,, \\
\tilde{h}_j &= \sum_{k \in C(j)} W_{hm} m_{jk} \,.
\end{aligned} \tag{4}$$

The gating units – input gate i, output gate o, and forget gate f – are computed in the same way as in the TreeLSTM with Eq. (1).[2]

Multiplicative TreeLSTM can be applied to any tree, where connection types between nodes are given. For example, in dependency trees, the semantic relations $r(j,k)$ between nodes are provided by a dependency parser.

4 Tree-Structured LSTMs with Abstract Meaning Representation

Tree-structured LSTMs have been applied successfully to syntactic parse trees (Tai et al., 2015; Miwa and Bansal, 2016). In this work, we look beyond *syntactic* properties of the text and incorporate *semantic* properties to the tree-structured LSTM model. Specifically, we utilize the network topology offered by a tree-structured LSTM and incorporate semantic features induced by AMR formalism. We aim to address the following questions: *In which tasks using AMR structures as the backbone for the tree-structured LSTM is useful? Furthermore, which semantic properties are useful for the given task?*

AMR is a semantic formalism where the meaning of a sentence is encoded as a single rooted, directed and acyclic graph (Banarescu et al., 2013). For example, the sentence "*A young girl is playing on the edge of a fountain and an older woman is not watching her*" is represented as:

[2] In the rest of the paper, we use the term *TreeLSTM* in a narrow sense to refer to the model corresponding to Eq. (1) and the term *tree-structured LSTM* to include both TreeLSTM and mTreeLSTM, unless specified otherwise.

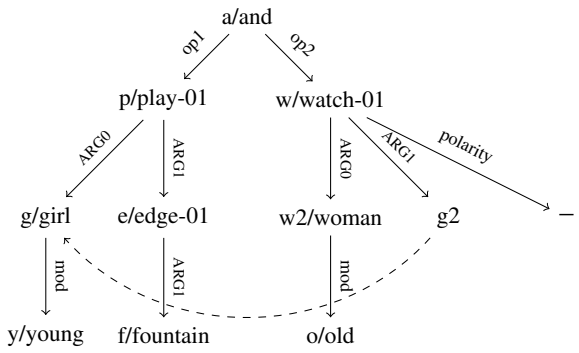

Figure 2: An AMR representing the sentence "A young girl is playing on the edge of a fountain and an older woman is not watching her".

```
(a / and
    :op1 (p / play-01
        :ARG0 (g / girl
            :mod (y / young))
        :ARG1 (e / edge-01
            :ARG1 (f / fountain)))
    :op2 (w / watch-01
        :ARG0 (w2 / woman
            :mod (o / old))
        :ARG1 g
        :polarity -))
```

The same AMR can be represented as in Figure 2, in which the nodes in the graph (also called concepts) map to words in the sentence and the edges represent the relations between words. AMR concepts consist of predicate senses, named entity annotations, and in some cases, simply lemmas of English words. AMR relations consist of core semantic roles drawn from the Propbank (Palmer et al., 2005) as well as fine-grained semantic relations defined specifically for AMR. Since AMR provides a whole-sentence semantic representation, it captures long-range dependencies among constituent words in a sentence. Similar to other semantic schemes, such as UCCA (Abend and Rappoport, 2013), GMB (Basile et al., 2012), UDS (White et al., 2016), AMR abstracts away from morphological and syntactic variability and generalize cross-linguistically.

To use AMR structures in a tree-structured LSTM, we first parse sentences to AMR graphs and transform the graphs to tree structures. The transformation follows the procedure used by Takase et al. (2016), splits the nodes with an indegree larger than one, which mainly present coreferential concepts, to a set of separate nodes, whose indegrees exactly equal one. We use JAMR (Flanigan et al., 2014, 2016), a statistical semantic parser trained on AMR bank, for AMR parsing.

On one hand, the AMR tree structure can be used directly with the TreeLSTM architecture described in Section 2, in which only node information is utilized to encode sentences into certain fixed-length embedding vectors. On the other hand, since AMR provides rich information about semantic relations between nodes, the mTreeLSTM architecture is more applicable due to its capability of modeling edges in the tree. We evaluate both encoded vectors produced by TreeLSTM and mTreeLSTM on AMR trees in Section 6.

5 Applications

In this section, we describe three specific models that apply the mTreeLSTM architecture and the AMR tree structures described above.

5.1 Sentiment Classification

In this task, we wish to predict the sentiment of sentences, in which two sub-tasks are considered: binary and fine-grained multiclass classification. In the former, sentences are classified into two classes (*positive* and *negative*), while in the latter they are classified into five classes (*very positive, positive, neutral, negative*, and *very negative*).

For a sentence x, we first apply tree-structured LSTMs over the sentence's parse tree to obtain the representation h_r at the root node r. A softmax classifier is then used to predict the class $\hat{y}$ of the sentence, with $\hat{p}_\theta\left(y \mid x\right) = \mathrm{softmax}\left(W^{(s)} h_r\right)$, where θ is the model parameters and $\hat{y} = \mathrm{argmax}_y\, \hat{p}_\theta\left(y \mid x\right)$. The cost function is the negative log-likelihood of the true sentiment class of the sentence with L2 regularization.

5.2 Semantic Relatedness

Given a sentence pair, the goal is to predict an integer-valued similarity score in $\{1, 2, ..., K\}$, where higher scores indicate greater degrees of similarity between the sentences.

Following Tai et al. (2015), we first produce semantic representation h_L and h_R for each sentence in the pair using the described models over each sentence's parse trees. Then, we predict the similarity score $\hat{y}$ using additional feedforward layers that consider a feature vector x_s consisting of both distance and angle between the pair (h_L, h_R): $\hat{p}_\theta = \mathrm{softmax}\left(W^{(p)}\, \sigma\left(W^{(s)} x_s\right)\right)$, $\hat{y} = r^\top \hat{p}_\theta$, where $r^\top = [1, 2, \ldots, K]$. Similar to Tai et al. (2015), we define a sparse target distribution p

such that the ground-truth rating $y \in [1, K]$ equals $r^\top p$ and use the regularized KL-divergence from $\hat{p}_\theta$ to p as the cost function.

5.3 Natural Language Inference (NLI)

In this task, the model reads two sentences (a premise and a hypothesis), and outputs a judgement of *entailment, contradiction*, or *neutral*, reflecting the relationship between the meanings of the two sentences.

Following Bowman et al. (2016), we frame the inference task as a sentence pair classification. First we produce representations h_P and h_H for the premise and hypothesis and then construct a feature vector x_c for the pair that consists of the concatenation of these two vectors, their difference, and their element-wise product. This feature vector is then passed to a feed-forward layer followed by a softmax layer to yield a distribution over the three classes: $\hat{p}_\theta = \text{softmax}\left(W^{(p)}\sigma\left(W^{(c)}x_c\right)\right)$. The negative log-likelihood of the true class labels for sentence pairs is used as the cost function.

6 Experiments

6.1 Hyperparameters and Training

The model parameters are optimized using AdaGrad (Duchi et al., 2011) with a learning rate of 0.05 for the first two tasks, and Adam (Kingma and Ba, 2015) with a learning rate of 0.001 for the NLI task. The batch size of 25 was used for all tasks and the model parameters were regularized with a per-minibatch L2 regularization strength of 10^{-4}. The sentiment and inference classifiers were additionally regularized using dropout with a dropout rate of 0.5.

Following Tai et al. (2015) and Zhu et al. (2015), we initialized the word embeddings with 300-dimensional GloVe vectors (Pennington et al., 2014). In addition, we use the aligner provided by JAMR parser to align the sentences with the AMR trees and then generate the embedding by using the GloVe vectors. The relation embeddings were randomly sampled from an uniform distribution in $[-0.05, 0.05]$ with a size of 100. The word and relation embeddings were updated during training with a learning rate of 0.1.

We use one hidden layer and the same dimensionality settings for sequential LSTM and tree-structured LSTMs. LSTM hidden states are of size 150. The output hidden size is 50 for the relatedness task and the NLI task. Each model is trained for 10 iterations. (We did not observe better results with more iterations.) The same training procedure repeats 5 times with parameters being evaluated at the end of every iteration on the development set. The model having the best results on the development set is used for final tests.

For all sentences in the datasets, we parse them with constituency parser (Klein and Manning, 2003), dependency parser (Chen and Manning, 2014), and AMR parser (Flanigan et al., 2014, 2016) to obtain the tree structures. We compare our mTreeLSTM model with two baselines: LSTM and TreeLSTM. We use the notation (C), (D), and (A) to denote the tree structures that the models are based on, where they stand for constituency trees, dependency trees, and AMR trees, respectively. The code to reproduce the results is available at `https://github.com/namkhanhtran/m-treelstm`.[3]

6.2 Sentiment Classification

For this task, we use the Stanford Sentiment Treebank (Socher et al., 2013b) with the standard train/dev/test splits of 6920/872/1821 for the binary classification sub-task, and 8544/1101/2210 for the fine-grained classification sub-task. We used two different settings for training: *root-level* and *phrase-level*. In the root-level setting, each sentence is a data point, while in the phrase-level setting, each phrase is reconstructed from nodes in the parse tree and treated as a separate data point. In the phrase-level setting we obtain much more data for training, but the root-level setting is closer to real-world applications. For AMR trees, we only report results in the root-level setting, as the annotation cost for the phrase-level setting is prohibitively high. We evaluate our models and baseline models at the sentence level.

Table 1 shows the main results for the sentiment classification task. While LSTM model obtains quite good performance in both settings, TreeLSTM model on constituency tree obtains better results, especially in the phrase level setting, which has more supervision. It confirms the conclusion from Tai et al. (2015) that combining linguistic knowledge with LSTM leads to better performance than sequence models in this task. Table 1 also shows mTreeLSTM consistently outperforms

[3]The correctness of our implementation is also suggested by the fact that we have reproduced the results of LSTM and TreeLSTM in Tai et al. (2015), up to small variations.

Model	Phrase-level		Root-level	
	5-class	2-class	5-class	2-class
LSTM	48.0 (1.0)	86.7 (0.7)	45.6 (1.1)	85.6 (0.5)
TreeLSTM(C)	**49.8** (0.8)	**87.9** (0.9)	46.3 (0.7)	**85.8** (0.5)
TreeLSTM(D)	46.9 (0.2)	85.5 (0.4)	46.0 (0.3)	85.0 (0.4)
TreeLSTM(A)	n/a	n/a	44.4 (0.2)	82.9 (0.6)
mTreeLSTM(A)	n/a	n/a	45.2 (0.5)	83.2 (0.5)
mTreeLSTM(D)	47.5 (0.7)	85.7 (0.1)	**46.7** (0.8)	85.7 (0.8)

Table 1: Accuracy on the Stanford Sentiment Treebank dataset with standard deviation in parentheses (numbers in percentage)

Model	Pearson	Spearman	MSE
LSTM	.841 (.004)	.778 (.006)	.304 (.003)
TreeLSTM (C)	.849 (.005)	.790 (.004)	.286 (.010)
TreeLSTM (D)	.863 (.003)	.803 (.002)	.260 (.005)
TreeLSTM (A)	.842 (.002)	.774 (.001)	.299 (.005)
mTreeLSTM (A)	.853 (.001)	.788 (.001)	.279 (.002)
mTreeLSTM (D)	**.872** (.004)	**.814** (.005)	**.244** (.007)

Table 2: Results on the SICK dataset for semantic relatedness task with standard deviation in parentheses

TreeLSTM on the same tree structures in both settings – Whenever a tree structure is applicable to both mTreeLSTM and TreeLSTM, the performance of mTreeLSTM with that tree structure is better. That is, in phrase-level setting, mTreeLSTM (D) outperforms TreeLSTM (D) and similarly in root-level setting, mTreeLSTM (D) and mTreeLSTM (A) perform better than TreeLSTM (D) and TreeLSTM (A), respectively. It demonstrates the effectiveness of the relation multiplication mechanism and the importance of modeling relation information. The TreeLSTM and mTreeLSTM models with AMR trees do not perform well on this task. Synthetic information along goes a long way in determining the sentiment of a sentence. Noisy sentences in this task also impact the accuracy of the AMR parser.

We now dive deep into what the models learn, by listing the composition matrices $W_{hh}^{r(j,k)}$ with the largest Frobenius norms. These matrices have learned larger weights, which are in turn being multiplied with the child hidden states. That child will therefore have more weight in the composed parent vector. In decreasing order of Frobenius norm, the relationship matrices for mTreeLSTM on dependency trees are: conjunction, adjectival modifier, object of a preposition, negation modifier, verbal modifier. The relationship matrices for mTreeLSTM on AMR trees are: negation (`:polarity`), attribute (`:ARG3`, `:ARG2`), modifier (`:mod`), conjunction (`:opN`). The model learns that verbal and adjective modifiers are more important than nouns, as they tend to affect the sentiment of sentences.

6.3 Sentence Relatedness

For this task, we use the Sentences Involving Compositional Knowledge (SICK) dataset, consisting of 9927 sentence pairs with the standard train/dev/test split of 4500/500/4927. Each pair is annotated with a relatedness score $y \in [1, 5]$, with 1 indicating the two sentences are completely unrelated, and 5 indicating they are very related. Following Tai et al. (2015), we use Pearson, Spearman correlations and mean squared error (MSE) as evaluation metrics.

Our results are summarized in Table 2. The tree-structured LSTMs, both TreeLSTM and mTreeLSTM, reach better performance than the standard LSTM. The model using dependency tree as the backbone achieves best results. The mTreeLSTM with AMR trees obtain slightly better results than the TreeLSTM with constituency trees. The multiplicative TreeLSTM models outperform the TreeLSTM models on the same parse trees, illustrating again the usefulness of incorporating relation information into the model.

Similar to the previous experiment, we list the composition matrices $W_{hh}^{r(j,k)}$ with the largest Frobenius norms. The relationship matrices for dependency trees include: indirect object, marker for introducing a finite clause subordinate to another clause, negation modifier, adjectival modifier, phrasal verb particle, conjunction. The relationship matrices for AMR trees are: patient (`:ARG1`), comparatives and superlatives (`:degree`), agent (`:ARG0`), attribute (`:ARG3`), medium (`:medium`), possession (`:poss`), manner (`:manner`).

6.4 Natural Language Inference

In this task, we first look at the SICK dataset described in the previous section. In this setting each sentence pair is classified into three labels, *entailment, contradiction*, and *neutral*.

In addition to the standard test set, we also report performances of our models on two different

Model	All	LS	Negation
LSTM	77.3 (0.5)	74.6 (1.4)	77.5 (0.4)
TreeLSTM (C)	79.0 (1.4)	78.1 (2.9)	85.3 (1.2)
TreeLSTM (D)	82.9 (0.3)	81.0 (2.6)	84.3 (1.2)
TreeLSTM (A)	82.6 (0.2)	84.0 (1.5)	88.2 (0.4)
mTreeLSTM (A)	83.3 (0.2)	**85.3** (0.4)	**88.5** (0.8)
mTreeLSTM (D)	**84.0** (0.5)	81.6 (1.3)	87.8 (0.8)

Table 3: Accuracy on the SICK dataset for the NLI task with standard deviation in parentheses (numbers in percentage)

Model	Acc (%)
LSTM (Bowman et al., 2015)	77.6
Syntax TreeLSTM (Yogatama et al., 2017)	80.5
CYK TreeLSTM (Maillard et al., 2017) *	81.6
Gumbel TreeLSTM (Choi et al., 2018)	81.8
Gumbel TreeLSTM + leaf LSTM (Choi et al., 2018)	82.6
TreeLSTM (D)	81.0
mTreeLSTM (D)	81.9

Table 4: Results on the SNLI dataset. The first group contains results of some best-performing tree-structured LSTM models on this data. (*: a preprint)

Model	rDim	# Params	Acc (%)
TreeLSTM (A)	n/a	301K	82.6
mTreeLSTM (A)	50	354K	82.7
mTreeLSTM (A)	75	358K	83.1
mTreeLSTM (A)	100	361K	83.6
mTreeLSTM (A)	200	376K	83.0

Table 5: Effects of the relation embedding size on SICK dataset for the NLI task

subsets. The first subset, *Long Sentence (LS)*, consists of sentence pairs in the test set where the premise sentence contains at least 18 words. We hypothesize that long sentences are more difficult to handle by sequential models as well as tree-structured models. The second subset, *Negation*, is a set of sentence pairs where negation words (*not*, *n't* or *no*) do not appear in the premise but appear in the hypothesis. In the test set, 58.7% of these examples are labeled as *contradiction*.

Table 3 summarizes the results of our models on different test sets. The mTreeLSTM models obtain highest results, followed by TreeLSTM models. The standard LSTM model does not work well on this task. The results reconfirm the benefit of using the structure information of sentences in learning semantic representations. In addition, Table 3 shows that TreeLSTM on dependency trees and AMR trees outperform the models with constituency trees. The dependency trees provide some semantic information, i.e., semantic relations between words at some degrees, while AMR trees present more semantic information. The multiplicative TreeLSTM on AMR trees perform much better than other models on the *LS* and *Negation* subsets. The results on the *LS* subset shows that mTreeLSTM on AMR trees can handle long-range dependencies in a sentence more effectively. For example, only mTreeLSTM (A) is able to predict the following example correctly:
Premise: *The grotto with a pink interior is being climbed by four middle eastern children, three girls and one boy.*
Hypothesis: *A group of kids is playing on a colorful structure.*
Label: *entailment*

Similar to previous experiments, we list the composition matrices with the largest Frobenius norms to get some insights into what the models learn. The relationship matrices for mTreeLSTM on dependency trees are: negation modifier, nominal subject, adjectival modifier, direct object, passive auxiliary, adverb modifier. These matrices for mTreeLSTM on AMR trees are: attribute (`:ARG2`), patient (`:ARG1`), conjunction (`:opN`), location, negation (`:polarity`), domain. In contrast to the sentiment classification task, where adjectives are crucial, the model learns that subjects and objects are important to determine the meaning of sentences.

Furthermore, we evaluate our mTreeLSTM model with SNLI (Stanford Natural Language Inference), a larger NLI dataset (Bowman et al., 2015). It is composed of about 550K/10K/10K sentence pairs in train/dev/test sets. We use dependency tree as the backbone for tree-structured LSTMs. All models in Table 4 use a hidden size of 100 for a fair comparison. The table shows that mTreeLSTM (D) outperforms many other syntax-based TreeLSTM models including TreeLSTM (D), reconfirming our conclusion drawn with SICK.

6.5 Additional Tests and Discussions

Incorporating relation information in the tree-structured LSTM increases model complexity. In this experiment, we analyze the impact of the dimensionality of relation embedding on the model

Model	# Params	Acc (%)
TreeLSTM (D)	301K	82.9
addTreeLSTM (D)	361K	83.4
fullTreeLSTM (D)	1.1M	83.5
mTreeLSTM (D)	361K	84.0

Table 6: Comparison between different methods using relation information on the SICK dataset for the NLI task

size and accuracy. Table 5 shows the model with the relation embedding size of 100 achieves the best accuracy, while the overall impact of the embedding size is mild. The multiplicative TreeLSTM has only 1.2 times the number of weights in TreeLSTM (with the same number of hidden units). We did not count the number of parameters in the embedding models since these parameters are the same for all models.

Table 6 shows a comparison between mTreeLSTM and two other plausible methods for integrating relation information with TreeLSTM. In *addTreeLSTM*, a relation is treated as an additional node input in the TreeLSTM model; In *fullTreeLSTM*, the model corresponds to Eq. (2), where each edge type has a separate transition matrix. Both models achieve better results than TreeLSTM, indicating the usefulness of relation information. While addTreeLSTM and fullTreeLSTM obtain comparable performances, mTreeLSTM outperforms both of them. It is also to note that the number of parameters of mTreeLSTM is much less than those of fullTreeLSTM.

7 Other Related Work

There is a line of research that extends the standard LSTM (Hochreiter and Schmidhuber, 1997) in order to model more complex structures. Tai et al. (2015) and Zhu et al. (2015) extended sequential LSTMs to tree-structured LSTMs by adding branching factors. They showed such extensions outperform competitive LSTM baselines on several tasks such as sentiment classification and semantic relatedness prediction (which is also confirmed in this paper). Li et al. (2015) further investigated the effectiveness of TreeLSTMs on various tasks and discussed when tree structures are necessary. Chen et al. (2017) combined sequential and tree-structured LSTM for NLI and has achieved state-of-the-art results on the benchmark dataset. Their approach uses *n-ary* TreeLSTM based on syntactic constituency parsers. In contrast, we focus more on child-sum TreeLSTM which is better suited for trees with high branching factor.

Previous works have studied the use of relation information. Dyer et al. (2015) considered each syntactic relation as an additional node and included its embedding to their composition function for dependency parsing. Peng et al. (2017) introduced a different set of parameters for each edge-type in their LSTM-based approach for relation extraction. In contrast to these works, our mTreeLSTM model incorporates relation information via a multiplicative mechanism, which we have shown is more effective and uses less parameters.

AMR has been successfully applied to a number of NLP tasks, besides the ones we considered in this paper. For example, Mitra and Baral (2016) made use of AMR to improve question answering; Liu et al. (2015) utilized AMR to produce promising results toward abstractive summarization. Using AMR as the backbone in TreeLSTM has been investigated in Takase et al. (2016). They incorporated AMR information by a neural encoder to the attention-based summarization method (Rush et al., 2015) and it performed well on headline generation. Our work differs from these studies in the sense that we aim to investigate how semantic information induced by AMR formalism can be incorporated to tree-structured LSTM models, and study which properties introduced by AMR turn out to be useful in various tasks. In this paper, we use the start-of-the-art AMR parser provided by Flanigan et al. (2016) which additionally provides the alignment between words and nodes in the tree.

Though we have considered AMR in this paper, we believe the conclusions we drew here largely apply to other semantic schemes, such as GMB and UCCA, as well. Abend and Rappoport (2017) has recently noted that the differences between these schemes are not critical, and the main distinguishing factors between them are their relation to syntax, their degree of universality, and the expertise they require from annotators.

8 Conclusions

We presented multiplicative TreeLSTM, an extension of existing tree-structured LSTMs to incorporate relation information between nodes in the tree. Multiplicative TreeLSTM allows different

compositional functions for child nodes, which makes it more expressive. In addition, we investigated how lexical semantic representation can be used with tree-structured LSTMs. Experiments on three common NLP tasks showed that multiplicative TreeLSTMs outperform conventional TreeLSTMs, illustrating the usefulness of relation information. Moreover, with AMR as backbone, tree-structured models can effectively handle long-range dependencies.

Acknowledgments

We would like to thank Matthias Seeger, Cedric Archambeau, Alex Klementiev, Ralf Herbrich, and anonymous reviewers for their helpful comments.

References

Omri Abend and Ari Rappoport. 2013. Universal conceptual cognitive annotation (UCCA). In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*. ACL, pages 228–238.

Omri Abend and Ari Rappoport. 2017. The state of the art in semantic representation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. ACL, pages 77–89.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop & Interoperability with Discourse*. pages 178–186.

Valerio Basile, Johan Bos, Kilian Evang, and Noortje Venhuizen. 2012. Developing a large semantically annotated corpus. In *Proceedings of the 8th Language Resources and Evaluation Conference*. LREC, pages 3196–3200.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. EMNLP, pages 632–642.

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

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. EMNLP, pages 740–750.

Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced LSTM for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. ACL, pages 1657–1668.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 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, pages 1724–1734.

Jihun Choi, Kang Min Yoo, and Sang goo Lee. 2018. Learning to compose task-specific tree structures. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI-18)*. AAAI.

Li Dong, Furu Wei, Chuanqi Tan, Duyu Tang, Ming Zhou, and Ke Xu. 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)*. ACL, pages 49–54.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12:2121–2159.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*. ACL, pages 334–343.

Jeffrey L. Elman. 1990. Finding structure in time. *Cognitive Science* 14:179–211.

Jeffrey Flanigan, Chris Dyer, Noah A. Smith, and Jaime Carbonell. 2016. CMU at SemEval-2016 task 8: Graph-based AMR parsing with infinite ramp loss. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*. SemEval, pages 1202–1206.

Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A. Smith. 2014. A discriminative graph-based parser for the abstract meaning representation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1)*. ACL, pages 1426–1436.

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

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1)*. ACL, pages 655–665.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. EMNLP, pages 1746–1751.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *The 3rd International Conference on Learning Representations*. ICLR.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*. ACL, pages 423–430.

Jiwei Li, Minh-Thang Luong, Dan Jurafsky, and Eduard Hovy. 2015. When are tree structures necessary for deep learning of representations? In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. EMNLP, pages 2304–2314.

Fei Liu, Jeffrey Flanigan, Sam Thomson, Norman M. Sadeh, and Noah A. Smith. 2015. Toward abstractive summarization using semantic representations. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*. NAACL.

Pengfei Liu, Xipeng Qiu, Yaqian Zhou, Jifan Chen, and Xuanjing Huang. 2016. Modelling interaction of sentence pair with coupled-LSTMs. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. EMNLP, pages 1703–1712.

Jean Maillard, Stephen Clark, and Dani Yogatama. 2017. Jointly learning sentence embeddings and syntax with unsupervised tree-LSTMs. *CoRR* .

Arindam Mitra and Chitta Baral. 2016. Addressing a question answering challenge by combining statistical methods with inductive rule learning and reasoning. In *Proceedings of the 13th AAAI Conference on Artificial Intelligence*. AAAI, pages 2779–2785.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using LSTMs on sequences and tree structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1)*. ACL, pages 1105–1116.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics* 31:71–106.

Nanyun Peng, Hoifung Poon, Chris Quirk, Kristina Toutanova, and Wen-tau Yih. 2017. Cross-sentence n-ary relation extraction with graph LSTMs. *Transactions of the Association for Computational Linguistics* pages 101–115.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. EMNLP, pages 1532–1543.

Alexander Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. EMNLP, pages 379–389.

Richard Socher, John Bauer, Christopher D. Manning, and Ng Andrew Y. 2013a. Parsing with compositional vector grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1)*. ACL, pages 455–465.

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

Richard Socher, Cliff C. Lin, Andrew Y. Ng, and Christopher D. Manning. 2011. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th International Conference on Machine Learning*. ICML, pages 129–136.

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

Ilya Sutskever, James Martens, and Geoffrey Hinton. 2011. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning*. ICML, pages 1017–1024.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*. ACL, pages 1556–1566.

Sho Takase, Jun Suzuki, Naoaki Okazaki, Tsutomu Hirao, and Masaaki Nagata. 2016. Neural headline generation on abstract meaning representation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. EMNLP, pages 1054–1059.

Aaron Steven White, Drew Reisinger, Keisuke Sakaguchi, Tim Vieira, Sheng Zhang, Rachel Rudinger, Kyle Rawlins, and Benjamin Van Durme. 2016. Universal decompositional semantics on universal dependencies. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. EMNLP, pages 1713–1723.

Dani Yogatama, Phil Blunsom, Chris Dyer, Edward Grefenstette, and Wang Ling. 2017. Learning to compose words into sentences with reinforcement learning. In *Proceedings of the 5th International Conference on Learning Representations*. ICLR.

Han Zhao, Zhengdong Lu, and Pascal Poupart. 2015. Self-adaptive hierarchical sentence model. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence*. IJCAI, pages 4069–4076.

Xiaodan Zhu, Parinaz Sobhani, and Hongyu Guo. 2015. Long short-term memory over recursive structures. In *Proceedings of the 32nd International Conference on Machine Learning*. ICML, pages 1604–1612.