Textual Inference with Tree-Structured LSTM

Adebayo Kolawole John$^{(\boxtimes)}$, Luigi Di Caro, Livio Robaldo, and Guido Boella

Dipartimento di Informatica, Universita Di Torino,
Corso Svizzera 185, 10149 Torino, Italy
kolawolejohn.adebayo@unibo.it, {dicaro,guido.boella}@di.unito.it,
livio.robaldo@uni.lu

Abstract. Textual Inference is a research trend in Natural Language Processing (NLP) that has recently received a lot of attention by the scientific community. Textual Entailment (TE) is a specific task in Textual Inference that aims at determining whether a hypothesis is entailed by a text. This paper employs the *Child-Sum Tree*-LSTM for solving the challenging problem of textual entailment. Our approach is simple and able to generalize well without excessive parameter optimization. Evaluation done on SNLI, SICK and other TE datasets shows the competitiveness of our approach.

Keywords: Child-Sum Tree LSTM · Information retrieval · Textual entailment

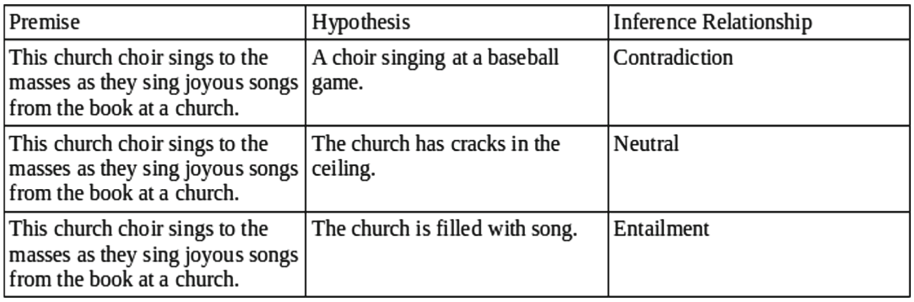
1 Introduction

Natural Language Inference (NLI) or put in another way, Textual Inference, refers to the process of identifying the type of logical/semantic relationship that exists between two texts. Since Dagan et al. [14] conceived the task of Recognizing Textual Entailment (RTE), it has continued to receive a lot of interest from researchers. To be specific, *entailment*, *contradiction* and *neutral* are examples of the inference relationships that are to be determined. Other examples of language inference tasks include text similarity [17], answer sentence selection [15], as well as Paraphrase Detection [28]. These tasks are challenging due to the prevalent variability and ambiguity in natural languages [13].

The preceeding work in NLI employs typical machine learning (ML) classifiers, this requires a lot of efforts for handcrafting feature for the classifiers. Moreover, these systems also rely on many language resources and tools, e.g., Semantic Nets etc. Also, the tiny size of the datasets involved, i.e., the SICK[1] and RTE[2] corpuses, which were the earliest datasets for RTE evaluation, contributes to the choice of ML classifiers because they have small training samples. This discourages the use of neural networks, which are more data-intensive. Most importantly, these systems rarely or slightly outperform simple baselines which rely on simple surface string similarity and word-overlap approaches [1,4].

1 http://clic.cimec.unitn.it/composes/sick.html.
2 https://www.aclweb.org/aclwiki/index.php?title=Textual_Entailment_Resource_Pool.

T. Bosse and B. Bredeweg (Eds.): BNAIC 2016, CCIS 765, pp. 17–31, 2017.
DOI: 10.1007/978-3-319-67468-1_2



Premise	Hypothesis	Inference Relationship
This church choir sings to the masses as they sing joyous songs from the book at a church.	A choir singing at a baseball game.	Contradiction
This church choir sings to the masses as they sing joyous songs from the book at a church.	The church has cracks in the ceiling.	Neutral
This church choir sings to the masses as they sing joyous songs from the book at a church.	The church is filled with song.	Entailment

Fig. 1. Sample texts from SNLI showing different classes of inference relationship

NLI is purely a classification task between different classes of inference relationship. Figure 1 shows one example for each of the three classes ‘contradiction’, ‘neutral’ and ‘entailment’ from the SNLI corpus[3].

Since the release of SNLI by Bowman et al. [9], a lot of research work that is based on neural networks have been published. Quite a good number of these systems are based on sentence encoding [9,11,26], where the Long Short-Term Memory (LSTM) networks have been used to embed premises and hypothesis in the same vector space. This enhances parameter sharing throughout the other components of a neural network model. Other techniques like the attention mechanism [20,21,23], extended memory structure [12,22] and factorization-based matching [32] builds on the former by providing more interaction between the embedded sentences.

Even though these systems have reported impressive result, they often exhibit a deep sentence modeling, which often translates to having excessive trainable parameters. Furthermore, the kind of interaction that the systems focus on downplays the syntactic relationship and interplay between the words in each sentence. However, we know that words do not live in isolation, and the meaning of a word is context dependent. Therefore, in order to know the true meaning of a word, it is also required that we know the meaning of the neighouring words that modifies its meaning.

This work employs a Tree-LSTM to obtain the representation of both the Premise[4] and its Hypothesis. Tree-LSTM is a linguistically intuitive choice for it readily captures the syntactic interpretations of a sentence structure [30]. Moreover, it combines the simplicity of the bag-of-word models with the order-sensitivity of the sequential models. Furthermore, similar to [32], we employ a matching scheme that parallelizes interaction between each child-node in the premise text to the nodes in the hypothesis text. Where, by nodes, we mean a dependency parsed representation of the texts under consideration. Our goal is to present a simple approach with fewer parameters and that does not require exces-

[3] http://nlp.stanford.edu/projects/snli.

[4] Throughout the paper, we use the words ‘Premise’, ‘Text’ or ‘First text’ interchangeably to mean the same thing, except otherwise specified.

sive parameter optimization, while being able to generalize well. The remaining parts of the paper are organized as follows. In the next section, we give a succinct review of some related work. Then, we describe our proposed method as well as the evaluation and result.

2 Related Work

The task of recognizing textual entailment (RTE) aims at making machines to mimic human inference capability, i.e., given a text P and a ground truth hypothesis Q, humans can easily recognize whether the meaning of Q can be directly inferred from P [14]. The goal of the RTE task is to design algorithms that replicate this human capability.

Bowman et al. [9] introduced the SNLI corpus, which contains 570 k human annotated text pairs. They used a lexical classifier as a baseline for their LSTM encoding-based network. Now, their approach is simple, both the premise and hypothesis were embedded in the same space using *Glove* vectors and then the sum of vectors of words in each sentence was used as the input to the LSTM. The authors in [26] proposed a word-by-word neural attention mechanism which is also based on LSTM. LSTM seems to be the natural choice because of its ability to retain information over many time-steps, although, Convolutional Neural Networks (CNN) has likewise shown to be a good choice for similar task [34].

In the work of Rocktaschel et al. [26], two LSTMs was used. While the first LSTM is reasoning over the sequence of tokens in the text, the other is performing the same computation on the hypothesis sequence. The second LSTM is conditioned by the output of the first one, i.e., its memory is initialized by the output (i.e., the last cell state) of the last hidden state of the first LSTM when reading each input from the hypothesis. An extension of their basic model instead utilized a Bi-directional LSTM, which was used in a similar manner in order to create a dual-attention. Baudis et al. [3] also reproduced an attention model similar to the question answering model in [31]. They utilized a Recurrent Neural Networks (RNN) model, a CNN model as well as a hybrid RNN-CNN model. The RNN captures long-term dependencies and contextual representation of words before being fed to the CNN.

Parikh et al. [23] improved on the attention mechanism of [26] by introducing two components, *Compare* and *Aggregate*. The former compares aligned phrases in the premise with that of the hypothesis and vice versa, using a feed-forward neural networks. The resulting vectors are summed over in the latter component, i.e. the *Aggregate* part. Cheng et al. [12] proposed a type of LSTM with enhanced memory, called the LSTMN which is similar to the memory networks of Wetson et al. [33]. They used two attention schemes, which they called the *Shallow fusion* and the *Deep fusion*. The *Shallow fusion* considers only the attention between the words in a text, which is usually called the intra-attention. The *Deep fusion* likewise performs the intra attention but much more, it tries to identify the importance of the words in the first text, in relation to the words in the second text. This is sometimes referred to as an inter-attention. The *Deep fusion* architecture utilizes both the inter-attention and intra-attention between the text and its corresponding hypothesis. Consequently, the *Deep fusion* architecture achieved a superior performance. Overall, both models achieved near state of the art results on the tasks of textual entailment, sentiment analysis.

Wang et al. [32] introduced a model which they called the Bilateral Multi-Perspective Matching model (BiMPM) for NLI. Their model obtained the state-of-the-art result when evaluated on the SNLI corpus. BiMPM share many commonalities with a few of the previous work which are already cited above. Specifically, Wang et al. [32] also utilized a Bi-directional LSTM. Now, a Bi-directional LSTM makes use of two LSTMs that are run in parallel. The first LSTM operates on the input sequence from the first time-step to the last time-step (forward) while the second LSTM operates on the same input sequence by performing computation from the last time-step to the first time-step (backward). Essentially, the hidden state of any time-step is the concatenation of both its forward and backward hidden states. Bi-directional LSTM is thus able to capture information in both context, i.e., the past and future information.

The innovative part of the work of Wang et al. [32] is how they utilized Bi-directional LSTM (BiLSTM) for the matching scheme that they proposed. First, a BiLSTM was used to separately encode the premise and the hypothesis. Then, using any of the four matching functions that they proposed, namely, the Full matching, Maxpooling matching, Attentive matching and the Max-attentive matching, each time-step of the premise is matched against every time-steps of the hypothesis and vice-versa. Another BiLSTM then combines the result before passing through a fully connected (FC) layer for classification. Apart from the choice of matching, the architecture remains the same. They obtained the best performance when the Full-Matching function was used.

Our proposed approach also utilizes a type of LSTM, i.e., the child-sum Tree LSTM for sentence encoding. The motivation for using this is that in a dependency parsing-based tree, because a headword incorporates information from each child, a natural intra-attention is created within each sentence, since the hidden vector of the headword is composed from those of its children. Furthermore, we also incorporate a high-level interaction scheme for the two texts, using a similar matching function to the one proposed in [32]. However, our approach is more simple and has fewer parameters. The Tree-structure LSTM network builds sentence representation from headword-child subphrases of a text, but takes into account more compositional features for better generalization.

3 Methods

We describe the general LSTM architecture. Specifically, this work employs the *Child-Sum Tree*-LSTMs proposed by [30]. We describe our inter attention scheme using a form of perspective matching [32].

Long Short-Term Memory Networks

Recurrent Neural Networks (RNNs) have connections that have loops, adding feedback and memory to the networks over time. This memory allows this type of network to learn and generalize across sequences of inputs rather than individual patterns. LSTM Networks [16] are a special type of RNNs and are trained using backpropagation through time, thus overcoming the vanishing gradient problem. LSTM networks have memory blocks that are connected into layers, the block contains gates that manage the blocks state and output. These gates are the *input* gates which decides the values from the input to update the memory state, the *forget* gates which decides what information to discard from the unit and the *output* gates which decides what to output based on input and the memory of the unit. LSTMs are thus able to memorize information over a long time-steps, since this information are stored in a recurrent hidden vector which is dependent on the immediate previous hidden vector. A unit operates upon an input sequence and each gate within a unit uses the sigmoid activation function to control whether they are triggered or not, making the change of state and addition of information flowing through the unit conditional.

At each time step t, let an LSTM unit be a collection of vectors in $\mathbb{R}^d$ where d is the memory dimension: an *input gate* i_t, a *forget gate* f_t, an *output gate* o_t, a *memory cell* c_t and a *hidden state* h_t. The state of any gate can either be open or closed, represented as [0,1]. The LSTM transition can be represented with the following equations (x_t is the an input vector at time step t, σ represents sigmoid activation function and $\odot$ the elementwise multiplication. The u_t is a tanh layer which creates a vector of new candidate values that could be added to the state):

$$\begin{aligned}
i_t &= \sigma\left(W^{(i)}x_t + U^{(i)}h_{t-1} + b^{(i)}\right),\\
f_t &= \sigma\left(W^{(f)}x_t + U^{(f)}h_{t-1} + b^{(f)}\right),\\
o_t &= \sigma\left(W^{(o)}x_t + U^{(o)}h_{t-1} + b^{(o)}\right),\\
u_t &= \tanh\left(W^{(u)}x_t + U^{(u)}h_{t-1} + b^{(u)}\right),\\
c_t &= i_t \odot u_t + f_t \odot c_{t-1},\\
h_t &= o_t \odot \tanh c_t \qquad (1)
\end{aligned}$$

Tree-Structured LSTM

Tree-LSTM is a specialized type of LSTM that adopt the tree-structure topology, i.e., at any given time step t,the LSTM is able to compose its states from an input vector and hidden states of its child-nodes simultaneously. This is unlike the standard LSTM that assumes a single child per unit, since the gating vectors and memory cell updates are dependent on the states of all child-nodes. Moreover, it maintains a forget gate separately for each child node. This characteristic enables the Tree-LSTM to be able to aggregate information from each child node. A good variant of the Tree-LSTM is the *Child-Sum Tree-LSTM*, which was proposed by Tai et al. [30]. The *Child-Sum Tree-LSTM* state transition is represented by the following equations, where C(j) is the set of the children of a node j, and k $\in$ C(j).

$$\begin{aligned}
\widehat{h_j} &= \sum_{k \in C(j)} h_k, \\
i_j &= \sigma\left(W^{(i)} x_j + U^{(i)} \widehat{h_j} + b^{(i)}\right), \\
f_{jk} &= \sigma\left(W^{(f)} x_j + U^{(f)} \widehat{h_k} + b^{(f)}\right), \\
o_j &= \sigma\left(W^{(o)} x_j + U^{(o)} \widehat{h_j} + b^{(o)}\right), \\
u_j &= \tanh\left(W^{(u)} x_j + U^{(u)} \widehat{h_j} + b^{(u)}\right), \\
c_j &= i_j \odot u_j + \sum_{k \in C(j)} f_{jk} \odot c_k, \\
h_j &= o_j \odot \tanh(c_j)
\end{aligned} \tag{2}$$

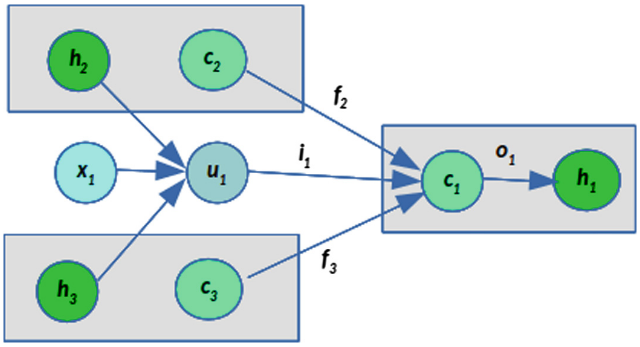


Fig. 2. Composing the memory cell c$_1$ and hidden state h$_1$ of a 2-children (subscripts 2 and 3) Tree-LSTM [30]

Tree-LSTM for Textual Entailment

We use the *Child-Sum Tree-LSTM* to generate sentence representation for both the text and hypothesis, taking as input the dependency-parse tree representation of the Premise and the Hypothesis texts. Giving a sentence, the structural connection between its constituent words form a deep branching graph, with elements and their dependencies where each connection in principle unites a head term and its dependent term(s). The dependent term maintains a one-to-one correspondence with its head, thus distinguishing between semantically useful words like nouns and verbs to say, a determinative word. With constituency parsing, a phrase-like one-to-one correspondence between the words is observed. The *Child-Sum Tree*-LSTM works better for dependency parse tree representation of a sentence, where each child is a node in the representation. For each node, the LSTM unit takes as input the vectors of its head word to which it is dependent. In the case of constituency parsing, an LSTM unit takes as input the exact vector of the node. Figure 2 shows how the hidden state and the memory cell for a Tree-LSTM unit with 2-children is composed. In the figure, labeled edges correspond to gating by the indicated gating vector, with dependencies omitted for compactness. The most important benefit of the Child-Sum Tree-LSTM is its discriminative capability, i.e., the model can learn parameters such that important words in the sentence are distinguished from unimportant words. This provides a natural and simple solution to what Attention mechanism [2] is used for.

Our approach can be divided into three parts, i.e., word encoding, sentence encoding and feature generation. Furthermore, we used two approaches for feature generation and classification, i.e., the distance based approximation and the perspective matching based approximation. The high-level representation of the two approaches is given in Figs. 3 and 4.

We assume two texts P = (p_1,....,p_i,....,p_M) and Q = (q_1,....,q_j,....,p_N) both of length M and N respectively. Also, a label y $\in$ Y is given, which shows the relationship, or put differently, the label for classification, e.g., entailment, neutral etc. For most of the datasets used in this work, y is either a binary output or a ternary output. P is typically the Premise or text and Q is the Hypothesis.

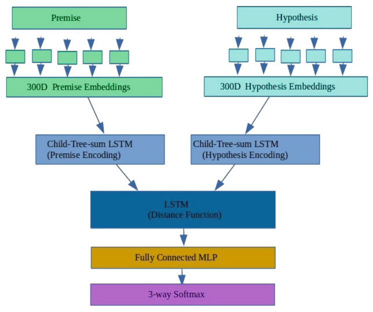

Premise
Hypothesis
300D Premise Embeddings
300D Hypothesis Embeddings
Child-Tree-sum LSTM (Premise Encoding)
Child-Tree-sum LSTM (Hypothesis Encoding)
LSTM (Distance Function)
Fully Connected MLP
3-way Softmax

Fig. 3. A high-level illustration of the Distance-based model

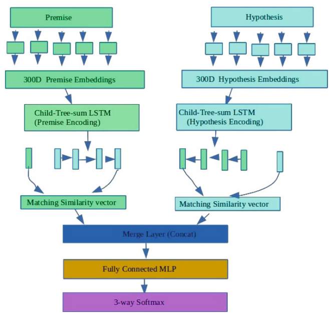

Premise
Hypothesis
300D Premise Embeddings
300D Hypothesis Embeddings
Child-Tree-sum LSTM (Premise Encoding)
Child-Tree-sum LSTM (Hypothesis Encoding)
Matching Similarity vector
Matching Similarity vector
Merge Layer (Concat)
Fully Connected MLP
3-way Softmax

Fig. 4. A high-level view of the Matching-based model

In anyway, the data representation follows the format: (P,Q,y) triples. The goal then, is to estimate the conditional probability Pr(y | P, Q) based on the training set, and predicting the relationship for testing samples by y* = $\text{argmax}_{y \in Y}$ Pr(y | P, Q).

Irrespective of the approach, we first encode each word with a BiLSTM and then obtain a sentential representation for both the Premise and the Hypothesis using a Child-Sum Tree-LSTM, which operates on a dependency parse tree representation of the texts.

Word Encoding

Here, we represent each word in the sentences P and Q with a d-dimensional vector, where the vectors are obtained from a word embedding matrix. Generally, we make use of the 300-dimensional *GLOVE* vectors, obtained from 840 billion words [24]. A Bi-directional LSTM is then used in order to obtain contextual information between the words. A Bi-directional LSTM is essentially composed of two LSTMs, one capturing information in one direction from the first time step to the last time-step while the other captures information from the last time-step to the first. The outputs of the two LSTMs are then combined to obtain a final representation which summarizes the information of the whole sentence. Equations (3) and (4) describes this computation.

$$\begin{aligned} \overrightarrow{h_i^p} &= \overrightarrow{LSTM}(\overrightarrow{h_{i-1}^p}, P_i), \quad i \in [1, ..., M] \\ \overleftarrow{h_i^p} &= \overleftarrow{LSTM}(\overleftarrow{h_{i-1}^p}, P_i), \quad i \in [M, ..., 1] \end{aligned} \tag{3}$$

$$h_i^f = [\overrightarrow{h_i^p}; \overleftarrow{h_i^p}] \tag{4}$$

Typically, when using an ordinary LSTM or BiLSTM to encode the words in a sentence, the whole sentence representation can be obtained as the final hidden state of the last word or time-step. The significant difference in our approach is that instead of using such final hidden state from the last time-step, we instead obtain the hidden states for each time-steps separately and then feed these hidden states into the Child-Sum Tree-LSTM. Analogously, we can also feed in the raw embedding vectors of each word into the Child-Sum Tree-LSTM, such that a dependent node takes the fixed raw vectors of its head while a head node takes the sum of its vectors and that of all its dependents. However, using the hidden states obtained when we first encode each word with a BiLSTM ensures that we have a context-aware high-level representation such that a dependent node takes the hidden state value of its head while a head node takes the sum of its hidden state along with that of all its dependents.

Consequently, we obtain the sentence representation for both P and Q, using the Child-Sum Tree-LSTM. An interesting thing is that with this setup, we do not require any form of Attention. Moreover, Attention is a way of focusing specially on some important parts of an input, and has been used extensively in some language modeling tasks [2,23]. Essentially, it is able to identify the parts of a text that are most important to the overall meaning of the text.

Distance Based Approximation

The idea here follows the work of [30]. A high level representation of this approach is shown in Fig. 3. First, we use the child-sum Tree-LSTM to encode sentences P and Q as explained in Sect. 3, to obtain the representations h_P and h_Q.

The representations h_P and h_Q can be regarded as a high level representation of both texts P and Q. Given h_P and h_Q, we predict the label $\hat{\mathrm{y}}_j$ by using a fully connected (FC) perceptron neural network which encodes the entailment relationship $\{\mathrm{r}\}_j = (\mathrm{h}_P, \mathrm{h}_Q)$ as the distance and angle between the element-wise summed vectors of the pair $(\mathrm{h}_P, \mathrm{h}_Q)$. We describe this process using Eqs. (5) and (9).

$$h_\times = h_P \odot h_Q \tag{5}$$

$$h_+ = h_P - h_Q \tag{6}$$

$$h_s = \sigma\left(W^{(\times)}h_\times + W^{(+)}h_+ + b^{(h)}\right) \tag{7}$$

$$\widehat{p}\theta(y|\{r\}_j) = softmax\left(W^{(p)}h_s + b^{(p)}\right) \tag{8}$$

$$\widehat{y_j} = \arg\max_y \widehat{p}\theta(y|\{r\}_j) \tag{9}$$

We trained our model with the negative log-likelihood of the true class labels $\mathrm{y}^{(k)}$ according to Eq. (10), where m is the size of the training sample and is an L2 norm regularization hyperparameter.

$$J(\theta) = -\frac{1}{m}\sum_{k=1}^{m} log\widehat{p}\theta(y^{(k)}|x^{(k)}) + \frac{\lambda}{2}||\theta||_2^2 \tag{10}$$

Matching-Based Approximation

The distance based approach assumes some form of intra-attention with the child-sum Tree LSTM representation. Inter-sentence attention schemes have proven very effective at various semantic inference tasks e.g. machine translation [2] and even NLI [21]. Similar to the Distance-based approximation, we use the Child-sum Tree-LSTM to obtain the sentence representation of the premise (P) and hypothesis (Q), after separately obtaining an annotation for each word by encoding with a BiLSTM. We then use a matching-function which is similar to the one proposed by Wang et al. [32]. The matching function creates a similarity interaction between two texts, i.e., from one text to another text, this eventually creates a kind of inter-sentence attention. Note that unlike in [32], we did not include Bi-LSTM to explicitly model the contextual relationship between words of each sentence, this is already amply captured by the Child-sum Tree-LSTM which we used to encode each sentence. Also, instead of using the multi-perspective cosine function, we utilized the conventional cosine similarity without an additional trainable parameter. The matching function work as explained below.

$$\overrightarrow{\boldsymbol{match_i}}^{forward} = sim(\overrightarrow{\boldsymbol{h_i}}^P, \overrightarrow{\boldsymbol{h_i}}^Q) \tag{11}$$

$$\overleftarrow{\boldsymbol{match_i}}^{backward} = sim(\overrightarrow{\boldsymbol{h_i}}^Q, \overrightarrow{\boldsymbol{h_i}}^P) \tag{12}$$

$$sim = cos(V1, V2) \tag{13}$$

Given two inputs P and Q, we represent an interaction (P→Q) by a forward pass and interaction (Q→P) by the backward pass. In the forward pass (*see* Eq. 11), we compare the time-step from the last hidden state of P to every time-steps of Q. Similarly, in the backward pass (*see* Eq. 12), the computation is done in a similar way. We compare the time-step of the hypothesis from the last hidden state of Q to each of the time-steps in P. For both forward and backward passes, the comparison is done by obtaining how similar the two vectors are, using the cosine similarity formula in Eq. (13). This matching function creates a form of interconnection from one-time-step to every other time-steps, thus yielding two vectors of similarity scores.

In the original full-matching method of [32], they compared each time-step from one text to every time-step in the other text. Furthermore, the comparison is done with a Bi-LSTM which makes the approach further computationally expensive. Here, we only compare the sentence representation of one sentence with each word in the other sentence and vice-versa. Also, for simplicity, we use the hidden state from the last time-step of a text as its encoding representation.

Once we obtain the similarity score vectors, i.e., S_p and S_q for P and Q respectively, we introduce a merge layer in order to concatenate the two vectors. The resulting vector is then passed to a fully connected Multilayer Perceptron (MLP) network to learn the entailment relationship. The predicted class is obtained from the probability distribution given in Eq. (14). In order to train our neural network, we use Multi-Class Cross-Entropy loss function, with 20% dropout regularization [29].

$$\hat{y} = H([s_p; s_q]) \tag{14}$$

$$\hat{y} = \arg\max_{y} y_{(i)} | x_{(i)} \tag{15}$$

We trained our model with the cross-entropy loss given in Eq. 16 where θ_F, θ_G, θ_H are parameters to be learned.

$$L(\theta_F, \theta_G, \theta_H) = \frac{1}{J} \sum_{j=1}^{J} \sum_{c=1}^{C} y_c^{(j)} \log \frac{\exp(\hat{y}_c)}{\sum_{c=1}^{C} \exp(\hat{y}_c)} \tag{16}$$

4 Evaluation

The RTE PASCAL challenge [14] is an important avenue for researchers to submit TE systems for public evaluation. We evaluated our system on the PASCAL RTE3 dataset which consists of 800 sentence pairs both for development and test set. The RTE3 dataset has only two classes, i.e., the entailment relation can either be true or false. The SEMEVAL track offering similarity and entailment tasks also make use of the SICK dataset[5]. SICK consists of 10000 sentence pairs annotated for use in both sentence similarity and 3-way entailment task. Finally we evaluated our system on the SNLI corpus [9] which is a big entailment dataset that is publicly available.

In the context of our ongoing work in the legal domain[6] [5,7,8], we evaluated our models on a legal dataset of textual entailment. The three datasets cited above contain sentences that are domain independent and thus have no technical jargons. Our goal is to see how our model would perform within the complex legal domain. Legal texts seem intuitive, because they have some peculiarities which set them apart from day-to-day texts, since they employ legislative terms. For instance, a sentence can have a reference to another sentence (e.g., an article) without any explicit link to its text from within the quoting text. Also, sentences are usually long with several clausal dependencies, that is notwithstanding of its inter and intra-sentential anaphora resolution complexity. We opined that a system that is able to achieve good result in this scenario would generalize well given other domain dependent texts.

We used the COLIEE dataset[7] which is a Question-Answering legal corpus made available in the context of COLIEE Legal Information Extraction/Retrieval challenge. Task 2 of the challenge addresses NLI task, such that, given a sentence and a Query, a system identifies if there is an entailment or not. We provide our evaluation result on the 2015 training and test sets.

Experiment

In this work, we used the Child-sum Tree-LSTM similar to the one proposed in [30] to encode the texts. We obtained dependency tree of both the text and the hypothesis using Stanford dependency parser [10]. To embed the training data, We used 300-dimensional Glove vectors [24]. Also, we keep the weights of the embeddings fixed and thus, not trainable.

Our model was implemented based on Keras[8]. For the COLIEE, SICK and RTE task, we used sigmoid for distributing output probability while we used softmax for SNLI with three classes. We apply a random *Dropout* of 0.2 throughout the models.

[5] http://clic.cimec.unitn.it/composes/sick.html.
[6] Specifically, the MIREL project: http://www.mirelproject.eu, which is drawn from our past project EUCases [6].
[7] http://webdocs.cs.ualberta.ca/~miyoung2/COLIEE2016.
[8] https://github.com/fchollet/keras.

We used a uniform batch-size of 25 in all the experiments excluding that of SNLI dataset, with the batch-size = 256. We used ADAM, a stochastic optimizer with learning rate set at 0.01 and a decay value of 1e-4. Moreover, we used early-stopping, in order to keep track of the point where the loss ceases to decrease after 4 epochs. This also helps to reduce over-fitting on the training set.

Tables 1, 2, 3 and 4 shows the evaluation result on the datasets that we used in our experiment. For Table 2, we used the results from Rocktaschel et al., [26], Baudis et al. [3] and Bowman et al. [9] as the baseline systems on SNLI and SICK respectively. For PASCAL-RTE3, we compare our models to some ML classifier baselines since there is no recent work which use similar deep learning approach on that dataset. For the result on COLIEE dataset in Table 4, we include the result reported by [18] on the same dataset. We can see that the performance of our model is near state-of-the-art, even though we slightly have fewer trainable parameters when compared with some of the baseline systems. Our model that is based on the matching-interaction approach seems to generally outperform our second model. The performance on SNLI corpus is very close to that of Wang et al. [32], even though the architecture of our model is simpler in theory.

Table 1. Evaluation on SNLI dataset

Model	k	$—\theta—_m$	Train	Test
LSTM [9]	100	220 k	84.8	77.6
Classifier [9]	-	-	**99.7**	78.2
Neural Attention [26]	100	250 k	85.3	83.5
NTI-SLSTM-LSTM encoders [22]	300	400 k	82.5	83.4
BiLSTM encoders with intra-attention [21]	600	2.8 m	85.9	85.0
LSTMN with deep attention fusion [12]	450	3.4 m	88.5	86.3
ESIM + 300D Syntactic TreeLSTM [11]	600	7.7 m	93.5	88.6
BiMPM Ensemble [32]	300	6.4 m	93.2	**88.8**
Tree-LSTM-Distance-Angle (**This Paper**)	300	560 k	87.3	84.1
Tree-LSTM-Matching (**This Paper**)	300	2.2 m	90.6	86.4

Table 2. Evaluation on SICK dataset

Model	SICK Train	SICK Test
attn1511 [3]	85.80	76.70
LSTM-RNN [9]	**99.90**	80.80
Tree-LSTM-Distance-Angle (**This Paper**)	85.10	76.00
Tree-LSTM-Full-Matching (**This Paper**)	95.60	**81.80**

Table 3. Evaluation on PASCAL-RTE3 dataset

Model	Train	Test
Wikipedia co-training [35]	-	57.25
Wiki + RTE-3 [35]	-	59.00
SVM [25]	-	66.37
Naive Bayes [25]	-	65.87
Sha et al. [27]	-	85.16
Tree-LSTM-Distance-Angle	**95.76**	86.90
Tree-LSTM-Full-Matching	93.80	**89.20**

Table 4. Evaluation on COLIEE 2015 Dataset

Model	Accuracy (%)
Convolutional Neural Network [19]	51.5
Convolutional Neural Network with TF-IDF [19]	53.0
Convolutional Neural Network with LSA [19]	54.5
Tree-LSTM-Distance-Angle (**This Paper**)	53.84
Tree-LSTM-Full-Matching (**This Paper**)	**57.40**

Discussion and Conclusions

In this paper, we described a *Child-Sum Tree*-LSTM model which obtained a good result on 3 well known textual entailment datasets. The results of our evaluation are given in Tables 1, 2, 3, and 4.

We noticed that the worst result was obtained on the COLIEE corpus. However, the COLIEE corpus, with less than 500 training samples may not be a good corpus for neural networks. Also, the uniqueness of the corpus, being legislative texts, may also be a factor. Furthermore, compared to SNLI and SICK, both the text and hypothesis in COLIEE are unusually long. Nevertheless, we report improved result to the baselines from [18].

In Table 1, we can see that our result is very close to the current state-of-the-art system [32], which has more than double trainable parameters, compared to ours. In Table 2, we see that we obtained the best result on the SICK dataset. Even though the RTE-3 dataset contains small training samples, still our models significantly outperformed the ML classifiers, i.e., SVM and Naive Bayes. In all the experiments, we obtained better accuracy with our interaction or matching-based approximation model than the one with the distance-based approximation model. This is because of the fact that entailment is not just a function of similarity between the texts being compared as in the case of the distance-based approximation model. Compared to all the benchmarked baselines, our approach is simple, requires no attention mechanism and has fewer trainable parameters. In the future, we would like to do a comparative qualitative analysis of our two models. Also, we would like to see if incorporating attention will impact the performance significantly.

Acknowledgments. Kolawole J. Adebayo has received funding from the Erasmus Mundus Joint International Doctoral (Ph.D.) programme in Law, Science and Technology. Luigi Di Caro and Guido Boella have received funding from the European Union's H2020 research and innovation programme under the grant agreement No 690974 for the project "MIREL: MIning and REasoning with Legal texts". Livio Robaldo has received funding from the European Union's H2020 research and innovation programme under the grant agreement No 661007 for the project "ProLeMAS: PROcessing LEgal language in normative Multi-Agent Systems".

References

1. Androutsopoulos, I., Malakasiotis, P.: A survey of paraphrasing and textual entailment methods. J. Artif. Intell. Res. **38**, 135–187 (2010)
2. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473 (2014)
3. Baudiš, P., Šedivỳ, J.: Sentence pair scoring: towards unified framework for text comprehension. arXiv preprint arXiv:1603.06127 (2016)
4. Bentivogli, L., Clark, P., Dagan, I., Giampiccolo, D.: The seventh pascal recognizing textual entailment challenge. In: Proceedings of TAC 2011 (2011)
5. Boella, G., Di Caro, L., Humphreys, L., Robaldo, L., Rossi, R., van der Torre, L.: Eunomos, a legal document and knowledge management system for the web to provide relevant, reliable and up-to-date information on the law. In: Artificial Intelligence and Law (2016, to appear)
6. Boella, G., Di Caro, L., Graziadei, M., Cupi, L., Salaroglio, C.E., Humphreys, L., Konstantinov, H., Marko, K., Robaldo, L., Ruffini, C. and Simov, K., Violato, A., Stroetmann, V.: Linking legal open data: Breaking the accessibility and language barrier in european legislation and case law. In: Proceedings of the 15th International Conference on Artificial Intelligence and Law, ICAIL 2015. ACM, New York (2015)
7. Boella, G., Di Caro, L., Rispoli, D., Robaldo, L.: A system for classifying multi-label text into EuroVoc. In: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Law, ICAIL 2013, pp. 239–240. ACM, New York (2013)
8. Boella, G., Di Caro, L., Robaldo, L.: Semantic relation extraction from legislative text using generalized syntactic dependencies and support vector machines. In: Morgenstern, L., Stefaneas, P., Lévy, F., Wyner, A., Paschke, A. (eds.) RuleML 2013. LNCS, vol. 8035, pp. 218–225. Springer, Heidelberg (2013). doi:10.1007/978-3-642-39617-5_20
9. Bowman, S. R., Angeli, G., Potts, C., Manning, C.D.: A large annotated corpus for learning natural language inference. arXiv preprint arXiv:1508.05326 (2015)
10. Chen, D., Manning, C.D.: A fast and accurate dependency parser using neural networks. In: EMNLP, pp. 740–750 (2014)
11. Chen, Q., Zhu, X., Ling, Z., Wei, S., Jiang, H.: Enhancing and combining sequential and tree lstm for natural language inference. arXiv preprint arXiv:1609.06038 (2016)
12. Cheng, J., Dong, L., Lapata, M.: Long short-term memory-networks for machine reading. arXiv preprint arXiv:1601.06733 (2016)
13. Dagan, I., Dolan, B., Magnini, B., Roth, D.: Recognizing textual entailment: rational, evaluation and approaches-erratum. Nat. Lang. Eng. **16**(01), 105–105 (2010)
14. Dagan, I., Glickman, O., Magnini, B.: The PASCAL recognising textual entailment challenge. In: Quiñonero-Candela, J., Dagan, I., Magnini, B., d'Alché-Buc, F. (eds.) MLCW 2005. LNCS, vol. 3944, pp. 177–190. Springer, Heidelberg (2006). doi:10.1007/11736790_9
15. Feng, M., Xiang, B., Glass, M.R., Wang, L., Zhou, B.: Applying deep learning to answer selection: a study and an open task. In: 2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU), pp. 813–820. IEEE (2015)
16. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
17. John, A.K., Di Caro, L., Boella, G.: Normas at semeval-2016 task 1: Semsim: a multi-feature approach to semantic text similarity. In: Proceedings of SemEval (2016)

18. Kim, M.Y., Xu, Y., Goebel, R.: A convolutional neural network in legal question answering (2015)
19. Kim, M.-Y., Xu, Y., Goebel, R.: Legal question answering using ranking SVM and syntactic/semantic similarity. In: Murata, T., Mineshima, K., Bekki, D. (eds.) JSAI-isAI 2014. LNCS, vol. 9067, pp. 244–258. Springer, Heidelberg (2015). doi:10.1007/978-3-662-48119-6_18
20. Liu, P., Qiu, X., Huang, X.: Modelling interaction of sentence pair with coupled-LSTMs. arXiv preprint arXiv:1605.05573 (2016)
21. Liu, Y., Sun, C., Lin, L., Wang, X.: Learning natural language inference using bidirectional LSTM model and inner-attention. arXiv preprint arXiv:1605.09090 (2016)
22. Munkhdalai, T., Yu, H.: Neural semantic encoders. arXiv preprint arXiv:1607.04315 (2016)
23. Parikh, A.P., Täckström, O., Das, D., Uszkoreit, J.: A decomposable attention model for natural language inference. arXiv preprint arXiv:1606.01933 (2016)
24. Pennington, J., Socher, R., Manning, C.D.: Glove: global vectors for word representation. In: EMNLP, vol. 14, pp. 1532–1543 (2014)
25. Gaona, M.A.R., Gelbukh, A., Bandyopadhyay, S.: Recognizing textual entailment using a machine learning approach. In: Advances in Soft Computing, pp. 177–185 (2010)
26. Rocktäschel, T., Grefenstette, E., Hermann, K.M., Kočiskỳ, T., Blunsom, P.: Reasoning about entailment with neural attention. arXiv preprint arXiv:1509.06664 (2015)
27. Sha, L., Li, S., Chang, B., Sui, Z., Jiang, T.: Recognizing textual entailment using probabilistic inference. In: EMNLP, pp. 1620–1625 (2015)
28. Socher, R., Huang, E., Pennin, J., Manning, C., Ng, A.: Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In: Advances in Neural Information Processing Systems, pp. 801–809 (2011)
29. Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**(1), 1929–1958 (2014)
30. Tai, K.S., Socher, R., Manning, C.D.: Improved semantic representations from tree-structured long short-term memory networks. arXiv preprint arXiv:1503.00075 (2015)
31. Tan, M., Xiang, B., Zhou, B.: LSTM-based deep learning models for non-factoid answer selection. arXiv preprint arXiv:1511.04108 (2015)
32. Wang, Z., Hamza, W., Florian, R.: Bilateral multi-perspective matching for natural language sentences. arXiv preprint arXiv:1702.03814 (2017)
33. Weston, J., Chopra, S., Bordes, A.: Memory networks. arXiv preprint arXiv:1410.3916 (2014)
34. Yin, W., Schütze, H., Xiang, B., Zhou, B.: Abcnn: attention-based convolutional neural network for modeling sentence pairs. arXiv preprint arXiv:1512.05193 (2015)
35. Zanzotto, F.M., Pennacchiotti, M.: Expanding textual entailment corpora from wikipedia using co-training. In: Proceedings of the COLING-Workshop on The Peoples Web Meets NLP: Collaboratively Constructed Semantic Resources, vol. 128 (2010)