The Forest Convolutional Network: Compositional Distributional Semantics with a Neural Chart and without Binarization

Phong Le and **Willem Zuidema**
Institute for Logic, Language and Computation
University of Amsterdam, the Netherlands
{p.le,zuidema}@uva.nl

Abstract

According to the principle of compositionality, the meaning of a sentence is computed from the meaning of its parts and the way they are syntactically combined. In practice, however, the syntactic structure is computed by automatic parsers which are far-from-perfect and not tuned to the specifics of the task. Current recursive neural network (RNN) approaches for computing sentence meaning therefore run into a number of practical difficulties, including the need to carefully select a parser appropriate for the task, deciding how and to what extent syntactic context modifies the semantic composition function, as well as on how to transform parse trees to conform to the branching settings (typically, binary branching) of the RNN. This paper introduces a new model, the Forest Convolutional Network, that avoids all of these challenges, by taking a parse forest as input, rather than a single tree, and by allowing arbitrary branching factors. We report improvements over the state-of-the-art in sentiment analysis and question classification.

1 Introduction

For many natural language processing tasks we need to compute meaning representations for sentences from meaning representations of words. In a recent line of research on 'recursive neural networks' (e.g., Socher et al. (2010)), both the word and sentence representations are vectors, and the word vectors ("embeddings") are borrowed from work in distributional semantics or neural language modelling. Sentence representations, in this approach, are computed by recursively applying a neural network that combines two vectors into one (typically according to the syntactic structure provided by an external parser). The network, which thus implements a 'composition function', is optimized for delivering sentence representations that support a given semantic task: sentiment analysis (Irsoy and Cardie, 2014; Le and Zuidema, 2015), paraphrase detection (Socher et al., 2011), semantic relatedness (Tai et al., 2015) etc. Studies with recursive neural networks have yielded promising results on a variety of such tasks.

In this paper, we represent a new recursive neural network architecture that fits squarely in this tradition, but aims to solve a number of difficulties that have arisen in existing work. In particular, the model we propose addresses three issues:

1. how to make the composition functions adaptive, in the sense that they operate adequately for the many different types of combinations (e.g., adjective-noun combinations are of a very different type than VP-PP combinations);

2. how to deal with different branching factors of nodes in the relevant syntactic trees (i.e., we want to avoid having to binarize syntactic trees,[1] but also do not want ternary productions to be completely independent from binary productions);

3. how to deal with uncertainty about the correct parse inside the neural architecture (i.e., we don't want to work with just the best or k-best parses for a sentence according to an external model, but receive an entire distribution over possible parsers).

[1] Eisner (2001, Chapter 2) shows that using flat rules is linguistically beneficial, "most crucially, a flat lexical entry corresponds to the local domain of a headword-the word together with all its semantic arguments and modifiers". From the computational perspective, flat rules make trees less deep, thus avoiding the vanishing gradient problem and capturing long range dependencies.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1155–1164,
Lisbon, Portugal, 17-21 September 2015.

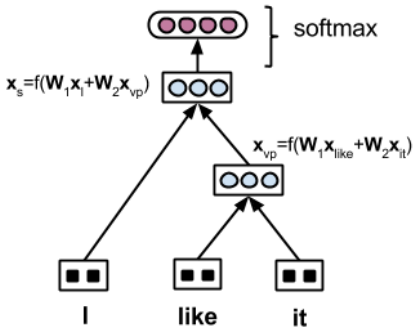

Figure 1: Recursive Neural Network. For simplicity, bias vectors are removed.

To solve these challenges we take inspiration from two other traditions: the convolutional neural networks and classic parsing algorithms based on dynamic programming. Including convolution in our network provides a direct solution for issue (2), and turns out, somewhat unexpectedly, to also provide a solution for issue (1). Introducing the chart representation from classic parsing into our architecture then allows us to tackle issue (3). The resulting model, the Forest Convolutional Network, outperforms all other models on a sentiment analysis and question classification task.

2 Background

This section is to introduce the recursive neural network (RNN) and convolutional neural network (CNN) models, on which our work is based.

2.1 Recursive Neural Network

A recursive neural network (RNN) (Goller and Küchler, 1996) is a feed-forward neural network where, given a tree structure, we recursively apply the same weight matrices at each inner node in a bottom-up manner. In order to see how an RNN works, consider the following example. Assume that there is a constituent with parse tree $(S\ I\ (VP\ like\ it))$ (Figure 1), and that $\mathbf{x}_I, \mathbf{x}_{like}, \mathbf{x}_{it} \in \mathbb{R}^d$ are the vectorial representations of the three words I, $like$ and it, respectively. We use a neural network which consists of a weight matrix $\mathbf{W}_1 \in \mathbb{R}^{d \times d}$ for left children and a weight matrix $\mathbf{W}_2 \in \mathbb{R}^{d \times d}$ for right children to compute the vector for a parent node in a bottom up manner. Thus, we compute $\mathbf{x}_{VP}$

$$\mathbf{x}_{VP} = f(\mathbf{W}_1\mathbf{x}_{like} + \mathbf{W}_2\mathbf{x}_{it} + \mathbf{b}) \quad (1)$$

where $\mathbf{b}$ is a bias vector and f is an (non-linear) activation function. Having computed $\mathbf{x}_{VP}$, we can then move one level up in the hierarchy and compute $\mathbf{x}_S$

$$\mathbf{x}_S = f(\mathbf{W}_1\mathbf{x}_I + \mathbf{W}_2\mathbf{x}_{VP} + \mathbf{b})$$

This process is continued until we reach the root node.

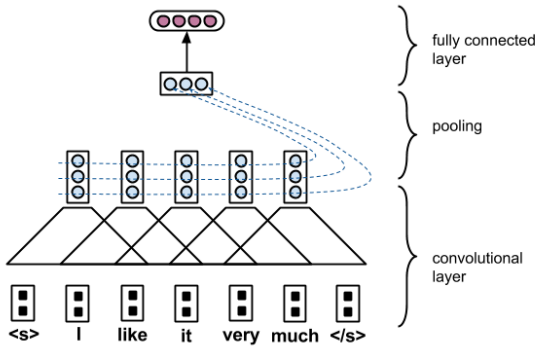

Figure 2: Convolutional Neural Network (one convolutional layer and one fully connected layer) with a window-size-3 kernel.

For classification tasks, we put a *softmax* layer on the top of the root node, and compute the probability of assigning a class c to an input $\mathbf{x}$ by

$$Pr(c|\mathbf{x}) = \text{softmax}(c) = \frac{e^{u(c,\mathbf{y}_{\text{top}})}}{\sum_{c' \in C} e^{u(c',\mathbf{y}_{\text{top}})}} \quad (2)$$

where $\left[u(c_1, \mathbf{y}_{top}), ..., u(c_{|C|}, \mathbf{y}_{top})\right]^T = \mathbf{W}_u\mathbf{y}_{top} + \mathbf{b}_u$; C is the set of all possible classes; $\mathbf{W}_u \in \mathbb{R}^{|C| \times d}, \mathbf{b}_u \in \mathbb{R}^{|C|}$ are a weight matrix and a bias vector.

Training an RNN uses the gradient descent method to minimize an objective function $J(\theta)$. The gradient $\partial J/\partial\theta$ is efficiently computed thanks to the back-propagation through structure algorithm (Goller and Küchler, 1996).

Departing from the original RNN model, many extensions have been proposed to enhance its compositionality (Socher et al., 2013; Irsoy and Cardie, 2014; Le and Zuidema, 2015) and applicability (Le and Zuidema, 2014b). The model we are going to propose can be considered as an extension of RNN with an ability to solve the three issues introduced in Section 1.

2.2 Convolutional Neural Network

A convolutional neural network (CNN) (LeCun et al., 1998) is also a feed-forward neural network; it consists of one or more convolutional layers (often with a pooling operation) followed by one or more fully connected layers. This architecture was invented for computer vision. It then has been widely applied to solve natural language processing tasks (Collobert et al., 2011; Kalchbrenner et al., 2014; Kim, 2014).

To illustrate how a CNN works, the following example uses a simplified model proposed by Collobert et al. (2011) which consists of one convolutional layer with the max pooling operation, followed by one fully connected layer (Figure 2). This CNN uses a *kernel* with window size 3; when we slide this kernel along the sentence "⟨s⟩ I like it very much ⟨/s⟩", we get five vectors:

$$\begin{aligned}
\mathbf{u}^{(1)} &= \mathbf{W}_1\mathbf{x}_{<s>} + \mathbf{W}_2\mathbf{x}_{I} + \mathbf{W}_3\mathbf{x}_{like} + \mathbf{b}_c \\
\mathbf{u}^{(2)} &= \mathbf{W}_1\mathbf{x}_{I} + \mathbf{W}_2\mathbf{x}_{like} + \mathbf{W}_3\mathbf{x}_{it} + \mathbf{b}_c \\
&\dots \\
\mathbf{u}^{(5)} &= \mathbf{W}_1\mathbf{x}_{very} + \mathbf{W}_2\mathbf{x}_{much} + \mathbf{W}_3\mathbf{x}_{</s>} + \mathbf{b}_c
\end{aligned}$$

where $\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3 \in \mathbb{R}^{d\times m}$ are weight matrices, $\mathbf{b}_c \in \mathbb{R}^m$ is a bias vector. The max pooling operation is then applied to those resulted vectors in an element-wise manner:

$$\mathbf{x} = \big[\max_{1\le i\le 5} \mathbf{u}_1^{(i)}, ..., \max_{1\le i\le 5} \mathbf{u}_j^{(i)}, ...\big]^T$$

Finally, a fully connected layer is employed

$$\mathbf{y} = f(\mathbf{W}\mathbf{x} + \mathbf{b})$$

where $\mathbf{W}, \mathbf{b}$ are a real weight matrix and bias vector, respectively; f is an activation function.

Intuitively, a window-size-k kernel extracts (local) features from k-grams, and is thus able to capture k-gram composition. The max pooling operation is for reducing dimension, forcing the network to discriminate important features from others by assigning high values to them. For instance, if the network is used for sentiment analysis, local features corresponding to k-grams containing the word "like" should receive high values in order to be propagated to the top layer.

3 Forest Convolutional Network

We now first propose a solution to the issues (1) and (2) (i.e., making the composition functions adaptive and dealing with different branching factors), called Recursive convolutional neural network (RCNN), and then a solution to the third issue (i.e., dealing with uncertainty about the correct parse), called Chart Neural Network (ChNN). A combination of them, Forest Convolutional Network (FCN), will be introduced lastly.

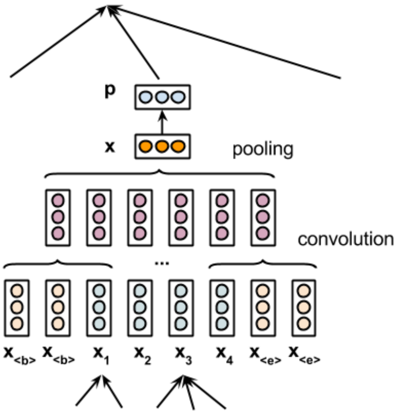
p
x
pooling
convolution
...
$x_{<b>}$ $x_{<b>}$ x_1 x_2 x_3 x_4 $x_{<e>}$ $x_{<e>}$

Figure 3: Recursive Convolutional Neural Network with a nonlinear window-size-3 kernel.

3.1 Recursive Convolutional Neural Network[2]

Given a subtree $p \rightarrow x_1 \ ... \ x_l$, an RCNN (Figure 3), like a CNN, slides a window-size-k kernel along the sequence of children $(x_1, ..., x_l)$ to compute a pool of vectors. The max pooling operation followed by a fully connected layer is then applied to this pool to compute a vector for the parent p.

This RCNN differs from the CNN introduced in Section 2.2 at two points. First, we use a non-linear kernel: after linearly transforming input vectors, an activation function is applied. Second, we put $k-1$ padding tokens <b> at the beginning of the children sequence and $k-1$ padding tokens <e> at the end. This thus guarantees that all the children contribute equally to the resulted vector pool, which now has $l+k-1$ vectors.

It is obvious that this RCNN can solve the second issue (i.e., dealing with different branching factors), we now show how it can make the composition functions adaptive. We first see what happens if the window size k is larger than the number of children l, for instance $k=3$ and $l=2$. There are four vectors in the pool

$$\begin{aligned}
\mathbf{u}^{(1)} &= f(\mathbf{W}_1\mathbf{x}_{<b>} + \mathbf{W}_2\mathbf{x}_{<b>} + \mathbf{W}_3\mathbf{x}_1 + \mathbf{b}_c) \\
\mathbf{u}^{(2)} &= f(\mathbf{W}_1\mathbf{x}_{<b>} + \mathbf{W}_2\mathbf{x}_1 + \mathbf{W}_3\mathbf{x}_2 + \mathbf{b}_c) \\
\mathbf{u}^{(3)} &= f(\mathbf{W}_1\mathbf{x}_1 + \mathbf{W}_2\mathbf{x}_2 + \mathbf{W}_3\mathbf{x}_{<e>} + \mathbf{b}_c) \\
\mathbf{u}^{(4)} &= f(\mathbf{W}_1\mathbf{x}_2 + \mathbf{W}_2\mathbf{x}_{<e>} + \mathbf{W}_3\mathbf{x}_{<e>} + \mathbf{b}_c)
\end{aligned}$$

where $\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3$ are weight matrices, $\mathbf{b}_c$ is a

[2] While finalizing the current paper we discovered a paper by Zhu et al. (2015) proposing a similar model which is evaluated on syntactic parsing. Our work goes substantially beyond theirs, however, as it takes a parse forest rather than a single tree as input.

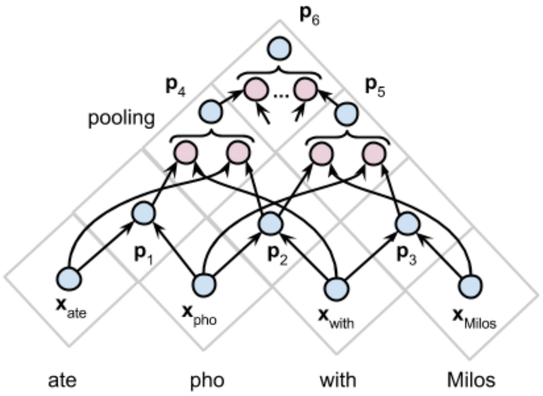

Figure 4: Chart Neural Network.

bias vector, f is an activation function. These four resulted vectors correspond to four ways of composing the two children:

(1) the first child stands alone (e.g., when the information of the second child is not important, it is better to ignore it),

(2,3) the two children are composed with two different weight matrix sets,

(4) the second child stands alone.

Now, imagine that we must handle binary syntactic rules with different head positions such as $S \rightarrow NP\ VP$ (e.g. "Jones runs") where the second child is the head and $VP \rightarrow VBD\ NP$ (e.g., "ate spaghetti") where the first child is the head. We can set those weight matrices such that when multiplying $\mathbf{W}_2$ by the vector of a head, we have a vector with high-value entries. And when multiplying $\mathbf{W}_2$ by the vector of a non-head, or when multiplying $\mathbf{W}_1$ or $\mathbf{W}_3$ by a vector, the resulted vector has low-value entries. This is possible thanks to the max pooling operation and that heads are often more informative than non-heads.

If the window size k is smaller than the number of children l, the argument above is still valid in some cases such as head position. However, there is no longer a direct interaction between any two children whose distance is larger than k.[3] In practice, this problem is not serious because rules with a large number of children are very rare.

3.2 Chart Neural Network

Unseen sentences are always parsed by an automatic parser, which is far from perfect and task-independent. Therefore, a good solution is to give the system a set of parses and let it decide which parse is the best or to combine some of them. The RNN model handles one extreme where this set contains only one parse. We now consider the other extreme where the set contains all possible parses. Because the number of all possible binary parse trees of a length-n sentence is the n-th Catalan number, processing individual parses is not practical. We thus propose a new model working on charts in the CKY style (Younger, 1967), called Chart Neural Network (ChNN).

We describe this model by the following example. Given a phrase "ate pho with Milos", a ChNN will process its parse chart as in Figure 4. Because any 2-word constituent has only one parse, the computation for p_1, p_2, p_3 is identical to Equation 1. For 3-word constituent p_4, because there are two possible productions $p_4 \rightarrow ate\ p_2$ and $p_4 \rightarrow p_1\ with$, we compute one vector for each production

$$\begin{aligned} \mathbf{u}^{(1)} &= f(\mathbf{W}_1\mathbf{x}_{ate} + \mathbf{W}_2\mathbf{p}_2 + \mathbf{b}) \\ \mathbf{u}^{(2)} &= f(\mathbf{W}_1\mathbf{p}_1 + \mathbf{W}_2\mathbf{x}_{with} + \mathbf{b}) \end{aligned} \tag{3}$$

and then apply the max pooling operation to these two vectors to compute $\mathbf{p}_4$. We do the same to compute $\mathbf{p}_5$. Finally, at the top, there are three productions $p_6 \rightarrow ate\ p_5$, $p_6 \rightarrow p_1\ p_3$ and $p_6 \rightarrow p_4\ Milos$. Similarly, we compute one vector for each production and employ the max pooling operation to compute $\mathbf{p}_6$.

Because this ChNN processes a chart like the CKY algorithm, its time complexity is $O(n^2d^2 + n^3d)$ where n and d are the sentence length and the dimension of vectors, respectively.[4] A ChNN is thus notably more complex than an RNN, whose complexity is $O(nd^2)$. Like chart parsing, the complexity can be reduced significantly by pruning the chart before applying the ChNN. This will be discussed right below.

3.3 Forest Convolutional Network

We now introduce the Forest Convolutional Network (FCN) model, which is a combination of the RCNN and the ChNN. The idea is to use an automatic parser to prune a chart[5], debinarize productions (if applicable), and then apply a ChNN

[3] An indirect interaction can be set up through pooling.

[4] In each cell, we apply the matrix-vector multiplication two times and (if the cell is not a leaf) apply the max pooling to a pool of maximally n d-D vectors.

[5] Pruning a chart by an automatic parser is also not perfect. However, the quality of a pruned chart can get very close to human annotation. For instance, the chart pruner proposed by Huang (2008) has a forest oracle of 97.8% F-

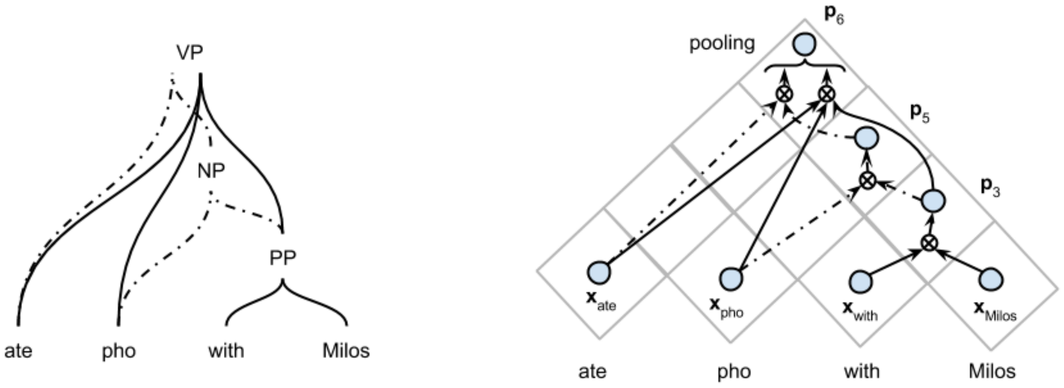

Figure 5: Forest of parses (left) and Forest Convolutional Network (right). $\otimes$ denotes a convolutional layer followed by the max pooling operation and a fully connected layer as in Figure 3.

where the computation in Equation 3 is replaced by a convolutional layer followed by the max pooling operation and a fully connected layer as in the RCNN.

Figure 5 shows an illustration how the FCN works on the phrase "ate pho with Milos". A forest of parses, given by an external parser, comprises two parses $(VP\ ate\ pho\ (PP\ with\ Milos))$ (solid lines) and $(VP\ ate\ (NP\ pho\ (PP\ with\ Milos)))$ (dash-dotted lines). The first parse is the preferred reading if Milos is a person, but the second one is a possible reading (for instance, if Milos is the name of a sauce). Instead of forcing the external parser to decide which one is correct, we let the FCN network do that because it has more information about the context and domain, which are embedded in training data. What the network should do is depicted in Figure 5-right.

Training Training an FCN is similar to training an RNN. We use the mini-batch gradient descent method to minimize an objective function J, which depends on which task this network is applied to. For instance, if the task is sentiment analysis, J is the cross-entropy over the training sentence set $\mathcal{D}$ plus an L2-norm regularization term

$$J(\theta) = -\frac{1}{|\mathcal{D}|}\sum_{s\in\mathcal{D}}\sum_{p\in s}\log Pr(c_p|\mathbf{p}) + \frac{\lambda}{2}||\theta||^2$$

where θ is the parameter set, c_p is the sentiment class of phrase p, $\mathbf{p}$ is the vector representation at the node covering p, $Pr(c_p|\mathbf{p})$ is computed by the softmax function, and λ is the regularization parameter.

The gradient $\partial J/\partial\theta$ is computed efficiently thanks to the back-propagation through structure (Goller and Küchler, 1996). We use the AdaGrad method (Duchi et al., 2011) to automatically update the learning rate for each parameter.

4 Experiments

We evaluate the FCN model with two tasks: question classification and sentiment analysis. The evaluation metric is the classification accuracy.

Our networks were initialized with the 300-D GloVe word embeddings trained on a corpus of 840B words[6] (Pennington et al., 2014). The initial values for a weight matrix were uniformly sampled from the symmetric interval $\left[-\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}\right]$ where n is the number of total input units. In each experiment, a development set was used to tune the model. We run the model ten times and chose the run with the highest performance on the development set. We employed early stopping: training is halted if performance on the development set does not improve after three consecutive epochs.

4.1 Sentiment Analysis

The Stanford Sentiment Treebank (SST)[7] (Socher et al., 2013) which consists of 5-way fine-grained sentiment labels (very negative, negative, neutral, positive, very positive) for 215,154 phrases of 11,855 sentences. We used the standard splitting: 8544 sentences for training, 1101 for development, and 2210 for testing. The average sentence length is 19.1. In addition, the treebank

score on section 23 of the Penn Treebank whereas resulted forests are very compact: the average number of hyperedges per forest is 123.1.

[6] http://nlp.stanford.edu/projects/GloVe/

[7] http://nlp.stanford.edu/sentiment/treebank.html

Model	Fine-grained	Binary
RNTN	45.7	85.4
CNN	48.0	88.1
DCNN	48.5	86.8
PV	48.7	87.8
DRNN	49.8	86.6
LSTM-RNN	49.9	88.0
CT-LSTM	**51.0**	88.0
FCN (dep.)	50.4	88.2
FCN (const.)	**51.0**	**89.1**

Table 1: Accuracies at sentence level on the SST dataset. FCN (dep.) and FCN (const.) denote the FCN with dependency forests and with constituent forests, respectively. The accuracies of RNTN, CNN, DCNN, PV, DRNN, LSTM-RNN and CT-LSTM are copied from the corresponding papers (see text).

also supports binary sentiment (positive, negative) classification by removing neutral labels, leading to: 6920 sentences for training, 872 for development, and 1821 for testing.

All sentences were parsed by Liang Huang's dependency parser[8] (Huang and Sagae, 2010). We used this parser because it generates parse forests and that dependency trees are less deep than constituent trees. In addition, because the SST was annotated in a constituency manner, we also employed the Charniak's constituent parser (Charniak and Johnson, 2005) with Huang (2008)'s forest pruner. We found that the beam width 16 for the dependency parser and the log probability beam 10 for the other worked best. Lower values harmed the system's performance and higher values were not beneficial.

Our FCN has the dimension of vectors at inner nodes 200, a window size for the convolutional kernel of 7, and the activation function $tanh$. It was trained with the learning rate 0.01, the regularization parameter 10^{-4}, and the mini batch size 5. To reduce the average depth of the network, the fully connected layer following the convolutional layer was removed (i.e., $\mathbf{p} = \mathbf{x}$, see Figure 3).

We compare the FCN against other models: the Recursive neural tensor network (RNTN) (Socher et al., 2013), the Convolutional neural network (CNN) (Kim, 2014), the Dynamic convolutional neural network (DCNN) (Kalchbrenner et al., 2014), the Paragraph vectors (PV) (Le and Mikolov, 2014), the Deep recursive neural network (DRNN) (Irsoy and Cardie, 2014), the Recursive neural network with Long short term memory (LSTM-RNN) (Le and Zuidema, 2015) and the Constituent Tree LSTM (CT-LSTM) (Tai et al., 2015).[9]

Table 1 shows the results. Our FCN using constituent forests achieved the highest accuracies in both fine-grained task and binary task, 51% and 89.1%. Comparing to CT-LSTM, although there is no difference in the fine-grained task, the difference in the binary task is significant (1.1%). Comparing to LSTM-RNN, the differences in both tasks are all remarkable (1.1% and 1.1%).

Constituent parsing is clearly more helpful than dependency parsing: the improvements that the FCN got are 0.6% in the fine-grained task and 0.9% in the binary task. We conjecture that, because sentences in the treebank were parsed by a constituent parser (here is the Stanford parser), training with constituent forests is easier.

4.2 Question Classification

In this task we used the TREC question dataset[10] (Li and Roth, 2002) which contains 5952 questions (5452 questions for training and 500 questions for testing). The task is to assign a question to one in six types: ABBREVIATION, ENTITY, DESCRIPTION, HUMAN, LOCATION, NUMERIC. The average length of the questions in the training set is 10.2 whereas in the test set is 7.5. This difference is due to the fact that those questions are from different sources. All questions were parsed by Liang Huang's dependency parser with the beam width 16.

We randomly picked 5% of the training set (272 questions) for validation. Our FCN has the dimension of vectors at inner nodes 200, a window size for the convolutional kernel of 5, and the activation function $tanh$. It was trained with the learning rate 0.01, the regularization parameter 10^{-4}, and the mini batch size 1. The vectors representing the two padding tokens <b>, <e> were fixed to $\mathbf{0}$.

We compare the FCN against the Convolutional neural network (CNN) (Kim, 2014), the Dynamic convolutional neural network (DCNN) (Kalch-

[8] http://acl.cs.qc.edu/~lhuang/software

[9] LSTM-RNN and CT-LSTM are very similar: they are RNNs using LSTMs for composition. Their difference is that LSTM-RNN uses one input gate for each child where as CT-LSTM uses only one input gate for all children.

[10] http://cogcomp.cs.illinois.edu/Data/QA/QC

Model	Acc. (%)
DCNN	93.0
MaxEnt$_H$	93.6
CNN-non-static	93.6
SVM$_S$	**95.0**
LSTM-RNN	93.4
FCN	94.8

Table 2: Accuracies on the TREC question type classification dataset. The accuracies of DCNN, MaxEnt$_H$, CNN-non-static, and SVM$_S$ are copied from the corresponding papers (see text).

brenner et al., 2014), MaxEnt$_H$ (Huang et al., 2008) (which uses MaxEnt with uni-bi-trigrams, POS, wh-word, head word, word shape, parser, hypernyms, WordNet) and the SVM$_S$ (Silva et al., 2011) (which uses SVM with, in addition to features used by MaxEnt$_H$, 60 hand-coded rules). We also include the LSTM-RNN (Le and Zuidema, 2015) whose accuracy was computed by running their published source code[11] on binary trees from the Stanford Parser[12] (Klein and Manning, 2003). This network was also initialized by the 300-D GloVe word embeddings.

Table 2 shows the results.[13] The FCN achieved the second best accuracy, only lightly lower than SVM$_S$ (0.2%). This is a promising result because our network used only parse forests, unsupervisedly pre-trained word embeddings whereas SVM$_S$ used heavily engineered resources. The difference between FCN and the third best is remarkable (1.2%). Interestingly, LSTM-RNN did not perform well on this dataset. This is likely because the questions are short and the parse trees quite shallow, such that the two problems that the LSTM was invented for (long range dependency and vanishing gradient) do not play much of a role.

4.3 Visualization

We visualize the charts we obtained in the sentiment analysis task as in Figure 6. To identify how important each cell is for determining the final vector at the root, we compute the number of features of each that are actually propagated all the way to the root in the successive max pooling operations. The circles in a graph are proportional to this number. Here, to make the contribution of each individual cell clearer, we have set the window size to 1 to avoid direct interactions between cells.

At the lexical level, we can see that the FCN can discriminate important words from the others. Two words "most" and "incoherent" are the key of the sentiment of this sentence: if one of them is replaced by another word (e.g. replacing "most" by "few" or "incoherent" by "coherent"), the sentiment will flip. The punctuation "." however also has a high contribution to the root. This happens to other charts as well. We conjecture that the network uses the vector of "." to store neutral features and propagate them to the root whenever it can not find more useful features in other vectors. Our future work is to examine this.

At the phrasal level, the network tends to group words in grammatical constituents, such as "most of the action setups", "are incoherent". Ill-formed constituents such as "of the action" and "incoherent ." receive little attention from the network.

Interestingly, we can see that the circle of "incoherent" is larger than the circles of any inner cells, suggesting that the network is able to make use of parses containing direct links from that word to the root. This is evidence that the network has an ability of selecting (or combining) parses that are beneficial to this sentiment analysis task.

5 Related Work

The idea that a composition function must be able to change its behaviour on the fly according to input vectors is explored by Socher et al. (2013), Le and Zuidema (2015), among others. The tensor in the former is multiplied with the vector representations of the phrases it is going to combine to define a composition function (a matrix) on the fly, and then multiplies again with these vectors to yield a compound representation. In the LSTM architecture of the latter, there is one input gate for each child in order to control how the vector of the child affects the composition at the parent node. Because the input gate is a function of the vector of the child, the composition function has an *infinite* number of behaviours. In this paper, we instead slide a kernel function along the sequence of children to generate different ways of composition. Although the number of behaviours is limited (and depends on the window size), it simultane-

[11] https://github.com/lephong/lstm-rnn

[12] http://nlp.stanford.edu/software/lex-parser.shtml

[13] While finalizing the current paper we discovered a paper by Ma et al. (2015) proposing a convolutional network model for dependency trees. They report a new state-of-the-art accuracy of 95.6%.

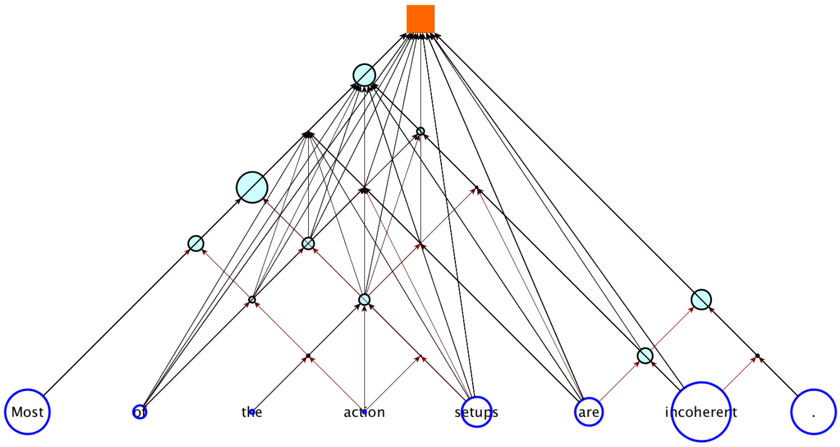

Figure 6: Chart of sentence "Most of the action setups are incoherent ." The size of a circle is propositional to the number of the cell's features that are propagated to the root.

ously provides us with a solution to handle rules with different branching sizes.

Some approaches try to overcome the problem of varying branching sizes. Le and Zuidema (2014b) use different sets of weight matrices for different branching sizes, thus requiring a large number of parameters. Because large branching-size rules are rare, many parameters are infrequently updated during training. Socher et al. (2014), for dependency trees, use a weight matrix for each relative position to the head word (e.g., first-left, second-right). Le and Zuidema (2014a) replace relative positions by dependency relations (e.g., OBJ, SUBJ). These approaches strongly depend on input parse trees and are very sensitive to parsing errors. The approach presented in this paper, on the other hand, does not need the information about the head word position and is less sensitive to parsing errors. Moreover, its number of parameters is independent from the maximal branching size.

Convolutional networks have been widely applied to solve natural language processing tasks. Collobert et al. (2011), Kalchbrenner et al. (2014), and Kim (2014) use convolutional networks to deal with varying length sequences. Recently, Zhu et al. (2015) and Ma et al. (2015) try to intergrate syntactic information by employing parse trees. Ma et al. (2015) extend the work of Kim (2014) by taking into acount dependency relations so that long range dependencies could be captured. The model proposed by Zhu et al. (2015), which is very similar to our Recursive convolutional neural network model, is to use a convolutional network for the composition purpose. Our work, although also employing a convolutional network and syntactic information, goes beyond them: we address the issue of how to deal with uncertainty about the correct parse inside the neural architecture. Therefore, instead of using a single parse, our proposed FCN model takes as input a forest of parses.

Related to our FCN is the Gated recursive convolutional neural network model proposed by Cho et al. (2014) which is stacking $n-1$ convolutional neural layers using a window-size-2 gated kernel (where n is the sentence length). Mapping their network into a chart, each cell is only connected to the two cells right below it. What makes this network special is the gated kernel which is a 3-gate switcher for choosing one of three options: directly transmit the left/right child's vector to the parent node, or compose the vectors of the two children. Thanks to this, the network can capture any binary parse trees by setting those gates properly. However, because only one gate is allowed to open in a cell, the network is not able to capture an arbitrary forest. Our FCN is thus more expressive and flexible than their model.

6 Conclusions

We proposed the Forest Convolutional Network (FCN) model that addresses the three issues: (1) how to make the composition functions adaptive, (2) how to deal with different branching factors of nodes in the relevant syntactic trees, (3) how to deal with uncertainty about the correct parse inside the neural architecture. The key principle is to carry out many different ways of computation and then choose or combine some of them. For more details, the two first issues are solved by employing a convolutional net for composition. To the third issue, the network takes input as a forest of parses instead of a single parse as in traditional approaches.

Our future work is to focus on how to choose/combine different ways of computation. For instance, we might replace the max pooling by different pooling operations such as mean pooling, k-max pooling (Kalchbrenner et al., 2014), and stochastic pooling (Zeiler and Fergus, 2013). We can even bias the selection/combination toward grammatical constituents by weighing cells by their inside probabilities.

Acknowledgments

We thank our three anonymous reviewers for their comments. This work was funded by the Faculty of Humanities of the University of Amsterdam, through a Digital Humanities fellowship to PL and a position in the New Generation Initiative (NGO) for WZ.

References

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 173–180. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder–decoder approaches. *Syntax, Semantics and Structure in Statistical Translation*, page 103.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.

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

Jason Eisner. 2001. *Smoothing a Probabilistic Lexicon via Syntactic Transformations*. Ph.D. thesis, University of Pennsylvania, July. 318 pages.

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

Liang Huang and Kenji Sagae. 2010. Dynamic programming for linear-time incremental parsing. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1077–1086. Association for Computational Linguistics.

Zhiheng Huang, Marcus Thint, and Zengchang Qin. 2008. Question classification using head words and their hypernyms. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 927–936. Association for Computational Linguistics.

Liang Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *ACL*, pages 586–594.

Ozan Irsoy and Claire Cardie. 2014. Deep recursive neural networks for compositionality in language. In *Advances in Neural Information Processing Systems*, pages 2096–2104.

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: Long Papers)*, pages 655–665, Baltimore, Maryland, June. Association for Computational Linguistics.

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

Dan Klein and Christopher D Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 423–430. Association for Computational Linguistics.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1188–1196.

Phong Le and Willem Zuidema. 2014a. The inside-outside recursive neural network model for dependency parsing. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Phong Le and Willem Zuidema. 2014b. Inside-outside semantics: A framework for neural models of semantic composition. In *NIPS 2014 Workshop on Deep Learning and Representation Learning*.

Phong Le and Willem Zuidema. 2015. Compositional distributional semantics with long short term memory. In *Proceedings of the Joint Conference on Lexical and Computational Semantics (*SEM)*. Association for Computational Linguistics.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics.

Mingbo Ma, Liang Huang, Bowen Zhou, and Bing Xiang. 2015. Dependency-based convolutional neural networks for sentence embedding. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 174–179, Beijing, China, July. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*, 12.

Joao Silva, Luísa Coheur, Ana Cristina Mendes, and Andreas Wichert. 2011. From symbolic to subsymbolic information in question classification. *Artificial Intelligence Review*, 35(2):137–154.

Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2010. Learning continuous phrase representations and syntactic parsing with recursive neural networks. In *Proceedings of the NIPS-2010 Deep Learning and Unsupervised Feature Learning Workshop*.

Richard Socher, Eric H. Huang, Jeffrey Pennington, Andrew Y. Ng, and Christopher D. Manning. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. *Advances in Neural Information Processing Systems*, 24:801–809.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings EMNLP*.

Richard Socher, Andrej Karpathy, Quoc V Le, Christopher D Manning, and Andrew Y Ng. 2014. Grounded compositional semantics for finding and describing images with sentences. *Transactions of the Association for Computational Linguistics*, 2:207–218.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566, Beijing, China, July. Association for Computational Linguistics.

Daniel H Younger. 1967. Recognition and parsing of context-free languages in time n 3. *Information and control*, 10(2):189–208.

Matthew D Zeiler and Rob Fergus. 2013. Stochastic pooling for regularization of deep convolutional neural networks. *arXiv preprint arXiv:1301.3557*.

Chenxi Zhu, Xipeng Qiu, Xinchi Chen, and Xuanjing Huang. 2015. A re-ranking model for dependency parser with recursive convolutional 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)*, pages 1159–1168, Beijing, China, July. Association for Computational Linguistics.