Sentiment Analysis with Deep Learning

Darshan Patel
Northeastern University

Abstract

Recently, deep learning based methods have shown significant improvement across different NLP tasks. This work compares various deep learning and deep learning inspired sentiment analysis methods for movie review. Along with sentiment classification, these methods are tackling the problem to learn better semantic vector representation for the phrases, sentences, and paragraphs. Results of the various flavor of Recursive, Recurrent, and Convolution Neural Networks are compared using Stanford Sentiment Treebank as the primary dataset and IMDB Large Movie Review dataset for both binary and fine-grained classification tasks.

1 Introduction

Sentiment analysis has become an import aspect as more and more businesses and individuals have started referring to the feedback of their customers and followers. Sentiment analysis is the task of finding the opinion/sentiment of the author by reviewing the piece of text. Sentiment analysis can be divided into four subtypes; 1) sentence-level 2) document-level 3) aspect-based 4) comparative sentiment analysis. In the sentence-level, the goal is to predict the sentiment of the small sentences like movie reviews. In the document-level, the goal is to predict the sentiment of a document such as a news article. In aspect-based, the goal is to identify the sentiment regarding the particular aspect of an item such as review of car fuel economy. In the comparative sentiment analysis, the goal is to determine the sentiment regarding an item by comparing it with another item.

The key problem for the mainly of the NLP domain to get the effective vector representation of the given text. Initially, methods such as one-hot encoding and bag of words were used to represent the text into the high dimensional vector. Despite their initial success, these methods are not able to include the semantic and morphological representation of the text into the vector representation.

Recent development in the field of deep learning has gained a tremendous amount of success in representing the word vector representation including semantic and morphological meaning. It started with unsupervised learning algorithm called as word2vec. Word2vec is an unsupervised algorithm, and it doesn't take context into account for learning vector representation. Various types of neural networks architecture are proposed to resolve this issue.

Firstly, recursive neural networks are proposed to include the contextual information in the high-dimensional vector representation of the text. All these model proposed supervised learning as they heavily depend on the parse tree structure of the given text. Section 2 explains and compares the various flavor of recursive neural networks.

The drawback of a recursive neural network is that they are a supervised algorithm, and there is not that many labeled data set available on the internet. To resolve that issue (Le and Mikolov et al., 2014) [6] proposed Paragraph Vectors, which is an unsupervised algorithm to learn the vector representation of the fixed length text. Section 3 thoroughly explains Paragraph Vectors.

Inspired by the success of convolution neural network in image processing, many researchers have explored the application of convolution neural network in the domain of Natural Language Processing. Section 4 discussion various convolution neural network architectures which are proposed for the learning the vector representation and sentiment analysis.

The final deep learning architecture is a recurrent neural network. Which is an effective method, given the input is a sequential data. The recurrent neural network can apply to natural language processing consider the sentence as the sequential data of the words. Section 5 discussion two flavors of the recurrent neural networks.

Most of the deep learning methods explained in this literature review are evaluated on the Stanford Sentiment Treebank dataset (Socher et al., 2013) [3]. This dataset contains the movie reviews extracted from *rottentomatoes.com* The dataset contains 10,662 sentences in which half of them are positive, and the other half are negative. This dataset also contains the sentiment label for all the 215,154 phrases of 10,662 sentences. These sentences also have five fine-grained labels such as "very negative", "negative", "neutral", "positive" and "very positive".

2 Recursive Neural Networks

2.1 Semi-Supervised Recursive Autoencoders (RAE)

Semantic representation of the text is a challenging topic in natural language processing domain. Traditionally approaches like one-hot encoding and bag-of-words have applied to represent it. However, they are not effectively to represent the semantic relationship between the text. (Socher et al., 2011) [1] has given such example where two sentences "white blood cells destroying an infection" and "an infection destroying white blood cells" have opposite semantic meaning but their traditional vector representations are same. (Socher et al., 2011) [1] proposed a Semi-Supervised Recursive Autoencoders (RAE) which can include semantic meaning in the vector representation. This method training can be unsupervised or semi-supervised.

Let's say an input word vector list $x = (x_1, x_2, x_3, x_4)$. Figure 1 shows the binary parse tree representation of these word vectors. Given the two word vectors $(c_1; c_2)$ as the inputs. The following equation can calculate the parent word vector

$$p = f(W^{(1)}[c_1; c_2] + b^1)$$

Here $W^{(1)}$ is a weight matrix and b^1 is a bias matrix; both are parameter matrix. These parameters can be learned by minimizing the following objective function.

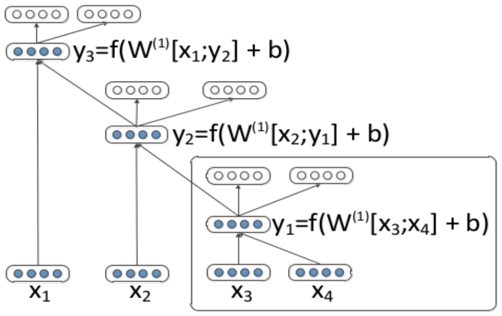


Figure 1: Binary tree of recursive autoencoder

$$E_{rec}([c_1; c_2]) = \frac{1}{2} ||[c_1; c_2] - [c'_1; c'_2]||^2 \; where \; [c'_1; c'_2] = W^{(2)}p + b^2$$

Here $[c'_1; c'_2]$ is a reconstruction of the child using the parent p using different weight and bias matrix. The proposed objective function is called as reconstruction error, which just compares the learnt representation and the actual representation of the word vectors.

In the unsupervised learning, parse tree of the text is not provided. In this setting, Greedy Unsupervised RAE is used to build the parse tree. In each iteration of Greedy Unsupervised RAE, it selects the word pair which has minimum reconstruction error and then that the parent vector representation will replace word pair. The same recursive greedy process continues until all word pairs are combined into one vector.

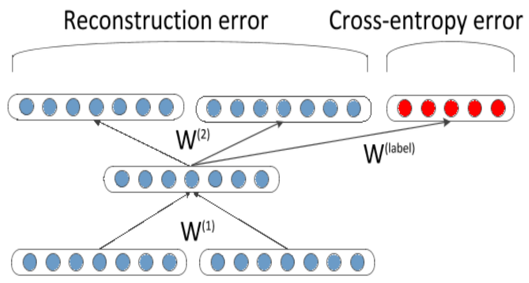


Figure 2: A semi-supervised RAE unit at a nonterminal tree node. Red nodes show the supervised softmax layer for label distribution prediction

To predict the sentence of phase-level target distribution, Greedy Unsupervised RAE can be extended to a semi-supervised setting. In the semi-supervised setting, target distribution vector will be attached to each node of the parse tree. A simple softmax layer is added, on top of each parent node, to predict the class distribution. Now the objective of semi-supervised RAE is to reduce the reconstruction error and cross-entry error at each parent node.

2.2 Recursive Matrix-Vector Spaces (MV-RNN)

Though Single-word vector space model mentioned in the previous section (Socher et al., 2011) [1] has been successful at learning lexical information, they are not able to capture the compositional meaning of the words. Example "extremely strong" cannot be obtained as the sum of word representation for "extremely" and "strong". (Socher et al., 2012) [2] has proposed a novel recursive neural network model for semantic compositionality. Figure 3 shows an illustration of the MV-RNN in which each a word or longer phrase has a matrix-vector (MV) representation. In which the vector captures the meaning of the word or longer phrase and the matrix captures how it modifies the meaning of the other words.

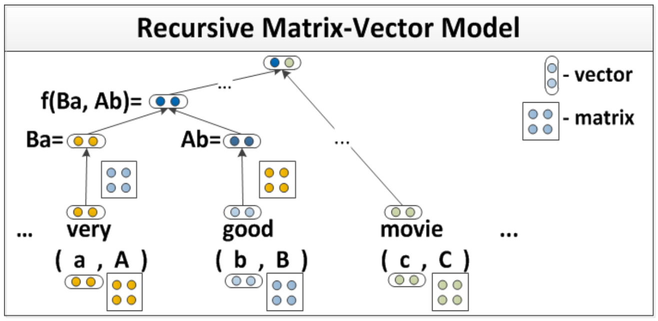


Figure 3: A recursive matrix-vector space neural network

The MV-RNN is one of the first models that can adequately negate sentiment when adjectives are combined with *not*. This model builds on a syntactically plausible parse tree and can handle compositional phenomena. MV-RNN model outperformed RAE model on a standard benchmark dataset of movie review (Socher et al., 2011) with 79% accuracy against 77.7% for classification on full-length movie review polarity.

2.3 Recursive Neural Tensor Network (RNTN)

The primary goal of MV-RNN is to capture the compositionality for better semantic representation. However, it cannot adequately capture the semantic of the longer phrases. There is a deep need of richer supervised training to capture the meaning of them. Apart from that, a number of parameters in MV-RNN becomes very large and depends on the size of the vocabulary. To resolve these issues, (Socher et al., 2013) [3] introduced the Stanford Sentiment Treebank – a movie review sentiment dataset with the fully labeled parse trees and Recursive Neural Tensor Network (RNTN). RNTN accurately predicts the long range compositional semantic effects using the labeled parse trees and a single powerful tensor composition function.

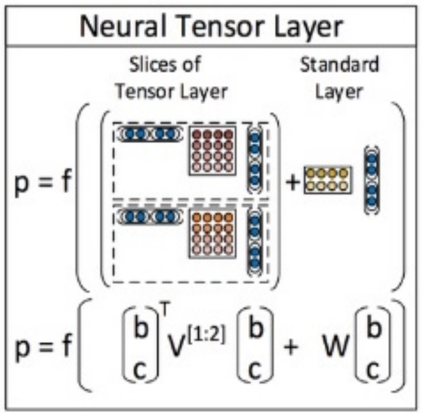


Figure 4: A single layer of the Recursive Neural Tensor Network

(See Figure 4) Here, b and c are vector representation of two words. V is the tensor that defines multiple bilinear forms – which captures the semantic compositionality. If below equation is compared with an equation in Figure 1 then it can be seen that the previous RNN model is a special case of the RNTN when V is set to 0.

$$p_1 = f(\begin{bmatrix} b \\ c \end{bmatrix}^T V^{[1:d]} \begin{bmatrix} b \\ c \end{bmatrix} + W \begin{bmatrix} b \\ c \end{bmatrix})$$

In RNTN, tensor V can directly relate the input vectors and capture a composition between them. In the paper, authors have done the model analysis on the sentences which has structures like "*X but Y*", Negating Positive Sentences and Negating Negative Sentences. Results show that the RNTN can accurately predict the sentiment of those sentences.

2.4 Deep Recursive Neural Networks (Deep RNN)

(Ozan Irsoy and Claire Cardie et al., 2014) [4] proposed a deep recursive neural network (Deep RNN) which built by stacking multiple recursive neural network layers. This idea inspired by the deep recurrent neural networks where multiple layers of a recurrent neural network are stacked. Multiple stacked layer in neural networks allows a layer to learn some part of compositional and pass the intermediate information to next successive layers.

In the previously proposed recursive neural networks, weights/tensors are shared between all the nodes of the parse tree. (Ozan Irsoy and Claire Cardie et al., 2014) [4] proposed that different weights should be considered for a leaf and an internal node. Because the matrix representation of the input node is dense as it contains vector representation of the words and matrix representation of the intermediate node is sparse as it derived by applying the activation function. It is called as untying leaves and internals. Thus, single weight matrix representation for the leaves and internal nodes is not sufficient as it has to represent both sparse and dense cases.

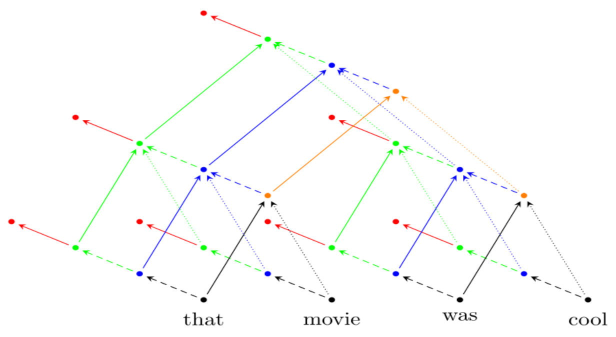


Figure 5: 3-layer Deep RNN. Output and input vectors represented as red and black points, respectively; Intermediate memory representations denoted by other colors.

In the stacked deep learners, a significant benefit is a hierarchy among hidden layers. Conceptually every hidden layer lies in the different space representation [5]. The below equation represents the concept of deep recursive neural network.

$$h^{(i)}{}_{\eta} = f(W^{(i)}{}_{L} h^{(i)}{}_{l(\eta)} + W^{(i)}{}_{R} h^{(i)}{}_{r(\eta)} + V^{(i)} h^{(i-1)}{}_{\eta} + b^{(i)})$$

Where i is the index of the multiple stacked layers, $W^{(i)}{}_{L}, W^{(i)}{}_{R}$ and $b^{(i)}$ are defined as parameters of each layer i and $V^{(i)}$ is the weight matrix which connects the $(i-1)$th hidden layer to the ith hidden layer. Deep

RNN has outperformed the previously purposed recursive neural networks on Stanford Sentiment Treebank binary and fine-grained classification task.

3 Paragraph Vectors

Paragraph Vector (Le and Mikolov et al., 2014) [6] is an unsupervised algorithm that learns the fixed-length vector representations of the variable length texts. The significant advantage of Paragraph Vectors over previously proposed supervised or semi-supervised recursive neural networks is that it doesn't depend on the parse trees not does it require word tuning for various tasks.

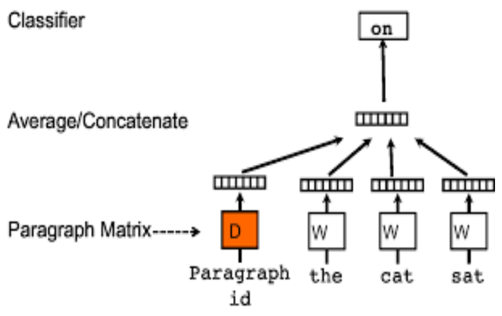


Figure 6: A framework for learning paragraph vector Distributed Memory Model of Paragraph vector(PV-DM)

The word2vec approach inspired the concept of Paragraph Vectors. In word2vec, initially, all word vectors contain random values but eventually they learn the representation of corresponding words. The same concept has applied to the paragraph vectors. In Paragraph Vectors, every paragraph is assigned to a unique vector and every word is also assigned to a unique vector both represented as a column in matrix D and W respectively. The paragraph vector considered as another word in the context. (See Figure 6)

Here Paragraph Vector performs the role of memory which remembers what is missing from the current context. Due to which it called as Distributed Memory Model of Paragraph Vector.

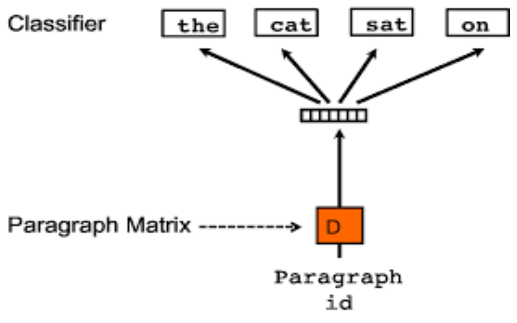


Figure 7: Distributed Bag of Words version of paragraph vector

Another approach is to learn the word vector representation from the given paragraph vector. (See Figure 7) In other words, during each step of training, sample the random words from the sampled text window and create a classification problem considering the Paragraph Vectors. This version of Paragraph Vector called as Distributed Bag of Words version of Paragraph Vector (PV-DBOW). For the Stanford Sentiment Treebank binary classification task, Paragraph Vectors outperformed all the previously proposed recursive neural network whereas Deep RNN performed better than Paragraph Vectors for the fine-grained classification task.

4 Convolution Neural Networks (CNN)

4.1 Deep Convolution Neural Network (Deep CNN)

Convolution Neural Network initially applied to image processing tasks, but its widespread has attracted NLP community. (Santos and Gatti et al., 2014) [7] proposed a deep convolution neural network for sentiment analysis of short texts. The model doesn't depend on external word vector representation of the texts; instead, it uses two convolution layers to learn the relevant features from characters, words and sentences of any size. It named as Character to Sentence Convolutional Neural Network (CharSCNN).

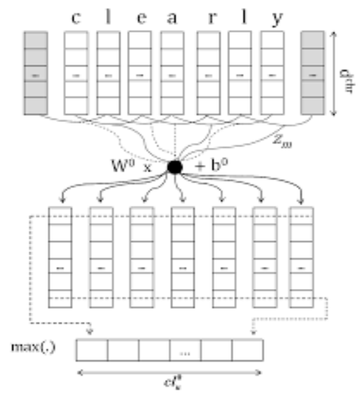


Figure 8: Convolution approach to character-level feature extraction

The first layer of the network learns the vector representation of the words. It captures the morphological, syntactic and semantic representation of the word phrases. Each word w_i converted into a vector $u_n = [r^{wrd}, r^{wch}]$ where r^{wrd} is a word-level embedding and r^{wch} is a character-level embedding. Parameters r^{wrd} *and* r^{wch} are learned by training the network.

Special treatment has given to character-level embedding. This model has applied a strategy proposed in (dos Santos and Zadrozny, 2014) [8] to learn the higher level word information using the characters of the word. As shown in Figure 8, (dos Santos and Zadrozny, 2014) [8] proposed the convolution approach to learn the local features around each character of the word and then a max operation applied to create a fixed-sized character-level embedding. Here window size is a hyper-parameter.

In the next layer, the same character level embedding approach is applied to learn the sentence-level representation. The authors used unsupervised pre-training of word-level embedding using word2vec's skip-gram method with a context window of size 9. This model has outperformed all the previously mentioned recursive neural network, but it didn't outperform Paragraph Vectors on Sanford Sentiment Treebank of binary and fine-grained classification task.

4.2 Dynamic Convolution Neural Network (Dynamic CNN)

(Kalchbrenner et al., 2014) [9] proposed Dynamic Convolution Neural Network for the semantic modeling of sentences. This model is a second successful application of convolution neural network for semantic content modeling.

Figure 9 shows the architecture of Dynamic Convolution Neural Network. The first layer of the model applies the wide convolution, which ensures that all weights of the filter reach the entire sentence. In the next layer, k-max pooling operation is performed on the feature maps generated by the first layer; where it selects the k-most powerful features and preserves the order of the features. The following equation selects the value of k.

$$k_l = \max\left(k_{top}, \left\lceil \frac{L-l}{L} s \right\rceil\right)$$

where l is the number of the current convolutional layer to which the max pooling is applied, k_{top} is the fixed pooling parameter for the topmost convolutional layer and L is the total number of convolution layers in the network;

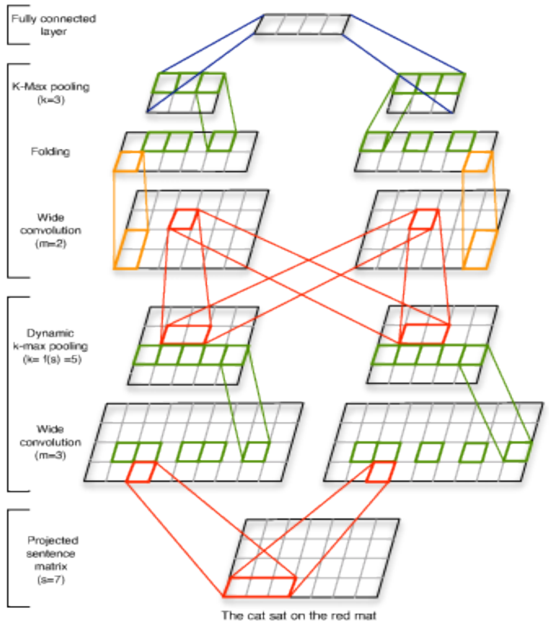

Fully connected layer
K-Max pooling (k=3)
Folding
Wide convolution (m=2)
Dynamic k-max pooling (k= f(s) =5)
Wide convolution (m=3)
Projected sentence matrix (s=7)
The cat sat on the red mat

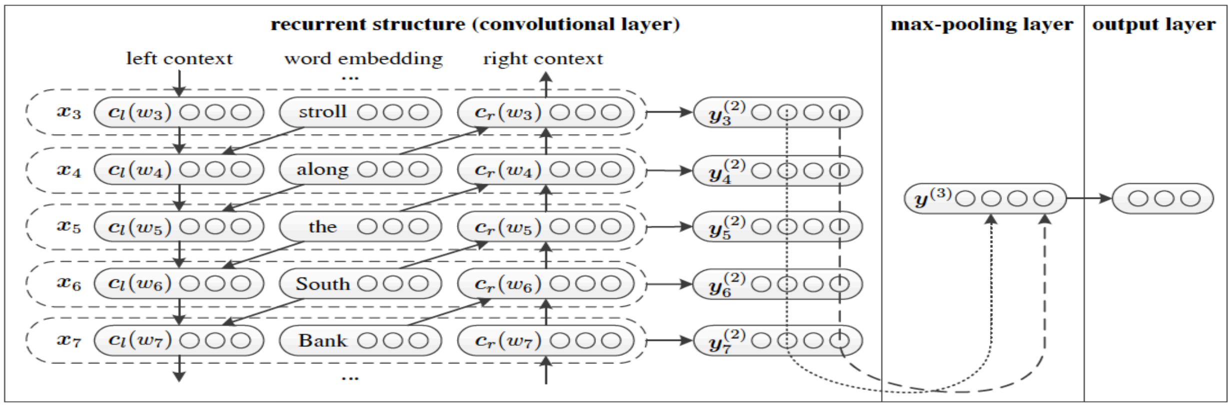
Figure 9: A DCNN for the seven-word input sentence.

After that layer, again wide convolution has used and then *folding* method has involved – which performs addition of every two rows in a feature map. For a feature map of d rows, folding returns a feature map of $d/2$ rows. Finally, k-max pooling and the fully connected layer has applied for the classification task. The author has not compared this model with the previously mentioned deep convolution neural networks. Apart from that, this model has outperformed all the previously mentioned recursive neural network, but it didn't outperform Paragraph Vectors on Sanford Sentiment Treebank of binary and fine-grained classification task.

5 Recurrent Neural Network

5.1 Recurrent Convolution Neural Network (RCNN)

Recurrent Neural Network analyzes a text word by word and stores the semantics of all the previous text in a fixed-size hidden layer. The advantage of Recurrent NN is its ability to capture the contextual information. However, RecurrentNN is a biased model, where following words are more dominant than earlier words. To tackle the bias problem, CNN an unbiased model is introduced in the NLP. However, CNN model proposed in the previous section uses the fixed window size, which is tough to determine. To solve this problem, (Lai, Xu, Liu and Zhao et al., 2015) [10] proposed a Recurrent Convolution Neural Network (RCNN).


recurrent structure (convolutional layer)
max-pooling layer
output layer
left context
word embedding
right context
x_3 $c_l(w_3)$ stroll $c_r(w_3)$ $y_3^{(2)}$
x_4 $c_l(w_4)$ along $c_r(w_4)$ $y_4^{(2)}$
x_5 $c_l(w_5)$ the $c_r(w_5)$ $y_5^{(2)}$
x_6 $c_l(w_6)$ South $c_r(w_6)$ $y_6^{(2)}$
x_7 $c_l(w_7)$ Bank $c_r(w_7)$ $y_7^{(2)}$
$y^{(3)}$
Figure 10: Structure of the recurrent convolutional neural network

This model combines a word and its surrounding contextual information to present a word. This model has a bidirectional recurrent neural network to capture the contextual information. Let's say $c_l(w_i)$ as the left context and $c_r(w_i)$ as the right context of word of word w_i. The following equations calculate $c_l(w_i)$ and $c_r(w_i)$. (See Figure 10)

$$c_l(w_i) = f\left(W^{(l)}c_l(w_{i-1}) + W^{(sl)}e(w_{i-1})\right) \ (equation\ 1)$$
$$c_r(w_i) = f\left(W^{(r)}c_r(w_{i+1}) + W^{(sr)}e(w_{i+1})\right) \ (equation\ 2)$$

Here $e(w_{i-1})$ is the word embedding of the previous word w_{i-1}, $c_l(w_{i-1})$ is the left side context of the previous word w_{i-1}. $W^{(l)}$ is a matrix which transforms the hidden layer (context) into the next hidden layer. $W^{(sl)}$ is a matrix which combines the semantic of current word with the next word's left context. The right side context $c_r(w_i)$ calculated in similar ways. Finally, each word is represented as $x_i = [c_l(w_i); e(w_i); c_r(w_i)]$. Once the representation of x_i obtained, linear transformation along with *tanh* activation function applied to get y_i. *which is* $y_i = \tanh(Wx_i + b)$. Then the pooling layer converts texts for various lengths into a fixed length vector and the last layer is a softmax function. When tested on Stanford Sentiment Treebank Recurrent CNN model are not as efficient as Paragraph Vectors.

5.2 Long Short-Term Memory (LSTM)

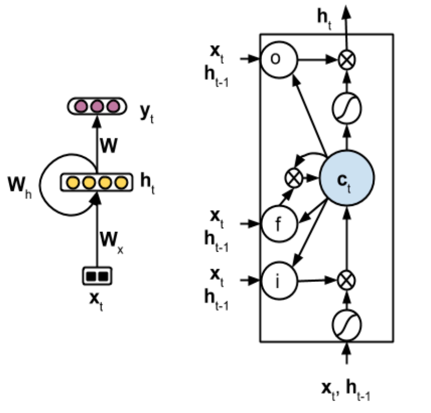


Figure 11: long short-term memory(right) and Recurrent neural network(left)

In practice, training recurrent neural networks with the gradient descent are challenging because gradient vanishes quickly after a few back-propagation steps. One solution to the problem is Long Short-Term Memory(LSTM) [12] [13]. The main idea of LSTM is to store the word vector information, present in the low parse tree into a memory cell by adding all the inputs and provide them much later step when it's needed. (See Figure 11) for comparison of LSTM and simple recurrent neural network architecture.

The same vanishing grading problem also happens when training recursive neural networks on deep trees. (Le and Zuidema et al., 2015) [11] proposed a model which modified recursive neural network using LSTM architecture to handle vanishing gradient problem. (See Figure 12)

Let's say there are two children nodes a and b and their parent is p. Based on the LSTM architecture the parent node p will have two input gates i_1, i_2 and two forget gates f_1, f_2 for the given two children. In logical sense, LSTM controls the effect of j^{th} child on memory unit using forget gate f_j and controls the effect of output at j^{th} child using input gate i_j

Traditional equation of recursive neural network is:

$$p_1 = g(W_1 x + W_2 y + b)$$

where b is a bias vector, W_1, W_2 are weight matrixes and g is an activation function

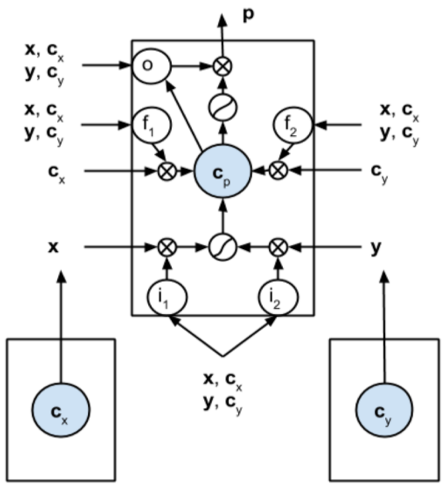


Figure 12: LSTM for recursive neural network

Using the LSTM with a recursive neural network, the function g is replaced by the proposed LSTM function (see Figure 12). This model also "untie" leaf nodes and inner nodes as mentioned in (Irsoy and Cardie et al., 2014). LSTM with recursive neural network outperformed all the previously mentioned model on Stanford Sentiment Treebank binary and fine-grained classification tasks.

5.3 Gated Recurrent Units (GRU)

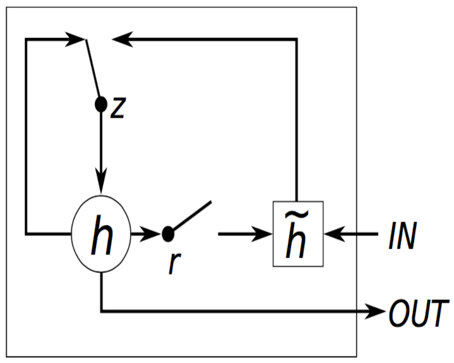


Figure 13: Structure of a gated recurrent unit

Gated Recurrent Units (GRU) is another solution to handle the vanishing gradient problem in recurrent neural networks. Gated Recurrent Units also has the ability to store the long range dependencies of the sentence. (Biswas, Chadda and Ahmad et al., 2015) [14] proposed a gated recurrent unit for sentiment analysis. They showed that GRUs were faster in convergence than LSTM and other gating networks.

Similar to the LSTM (Figure 11), the GRU (Figure 12) has gating units to control the flow of information inside the unit. However, GRU doesn't have memory cells. IN GRU, Update and reset gates control the flow of the information through hidden units.

For text classification using GRUs (See Figure 14), first words are encoded into vector representation using the 1 of K encoding. Word order also maintained in this encoding. The next layer, word embeddings takes these encoded vector representation as input and learn to represent them using the real-valued vector. These real-valued vectors are weights between the GRU hidden layer and word embeddings layer. The output of GRU hidden layer is vector representation of the given sequence of words, which can be used as input to any classifier. GRUs compared with LSTM using 50,000 IMDB movie reviews [15] and GRU outperformed traditional LSTM by 0.97 against 0.957 accuracy.

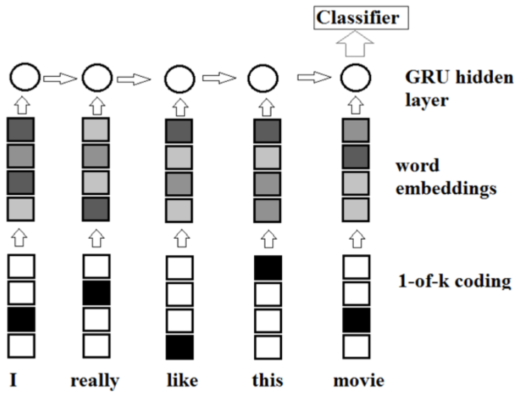


Figure 14: Text classification using GRU net

6 Model Evaluation

Model	Stanford Sentiment Treebank	
	Binary (%)	Fine-grained(%)
Naïve Bayes	81.8	41.0
Support Vector Machine(SVM)	79.4	40.7
Recursive Neural Networks (RNN)	82.4	43.2
Matrix-Vector RNN	82.9	44.4
Recursive Neural Tensor Network	85.4	45.7
Paragraph Vector	87.8	48.7
Deep RNN	86.6	49.8
Deep CNN	85.7	48.3
Dynamic CNN	86.8	48.5
Recurrent CNN	--	47.21
LSTM-RNN	**88.0**	**49.9**

Table 1: Performance evaluation of various Deep Learning models

7 Discussion and Conclusion

Long short-term Memory with the recursive neural network has gained the highest accuracy on the Stanford Sentiment Treebank for both binary and fine-grained classification tasks among all the discussion model. The open comparison question is how Gated Recurrent Units will perform on Stanford Sentiment Treebank. (Biswas, Chadda and Ahmad et al., 2015) [14] has compared GRU with LSTM and showed that GRU is effective than LSTM, but it would be interesting to see how much accuracy a model with a combination of GRU and the recursive neural network can achieve?

Apart from that, various flavors of convolution neural network architecture didn't perform well on the Stanford Sentiment Treebank, but Recurrent CNN is a very efficient model when input sentences are longer. ((Lai, Xu, Liu and Zhao et al., 2015) [10]). It's because recurrent functionality can efficiently handle the long-range semantic dependencies.

Effective semantic vector representation is a central point of all the models. Different models are trying different ways to relate the words to learn the meaning of their compositionality and semantics. It would be interesting to see how newly proposed successor of the recurrent neural network; Memory Network [16] will perform to learn the semantic vector representation?

References:

1. R. Socher, J. Pennington, E. H. Huang, A. Y. Ng, and C. D. Manning. 2011. Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions. In EMNLP.
2. R. Socher, B. Huval, C. D. Manning, and A. Y. Ng. 2012. Semantic compositionality through recursive Matrix-Vector spaces. In EMNLP.
3. Socher, Richard, Perelygin, Alex,Wu, Jean Y., Chuang, Jason, Manning, Christopher D., Ng, Andrew Y., and Potts, Christopher. Recursive deep models for semantic compositionality over a sentiment treebank. In Conference on Empirical Methods in Natural Language Processing, 2013.
4. Irsoy, Ozan and Cardie, Claire. Deep Recursive Neural Networks for Compositionality in Language, 2014. In NIPS.
5. Yoshua Bengio. Learning deep architectures for AI. Foundations and Trends in Machine Learning, 2(1):1–127, 2009.
6. 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.
7. Cicero Nogueira dos Santos and Maira Gatti, Brazilian IBM Research Lab. Deep Convolution al Neural Networks for Sentiment Analysis of Short Texts. 2014
8. Cicero Dos Santos and Bianca Zadrozny. Learning Character-level Representations for Part-of Speech Tagging Proceedings of The 31st International Conference on Machine Learning, pp. 1818–1826, 2014
9. N. Kalchbrenner, E. Grefenstette, P. Blunsom. 2014. A Convolutional Neural Network for Modelling Sentences. In ACL 2014.
10. Siwei Lai, Liheng Xu, Kang Liu and Jun zhao 2015. Recurrent Convolutional Neural Networks for Text Classification. In AAAI
11. Phong Le and Willem Zuidema. 2015. Compositional Distributional Semantics with Long Short Term Memory
12. Sepp Hochreiter and J¨urgen Schmidhuber. 1997. Long short-term memory. Neural computation, 9(8):1735– 1780.
13. Felix Gers. 2001. Long short-term memory in recurrent neural networks. Unpublished PhD dissertation, E´cole Polytechnique Fe´de´rale de Lausanne, Lausanne, Switzerland
14. Shamim Biswas, Ekamber Chadda and Faiyaz Ahmad. 2015. Sentiment Analysis with Gated Recurrent Units
15. Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts (2011) Learning Word Vectors for Sentiment Analysis
16. Jason Weston, Sumit Chopra & Antoine Bordes. Facebook AI Research. 2015. MEMORY NETWORKS