Feature Weight Tuning for Recursive Neural Networks

Jiwei Li
Computer Science Department
Stanford University
Stanford, CA, USA 94305
jiweil@stanford.edu

Abstract

This paper addresses how a recursive neural network model can automatically leave out useless information and emphasize important evidence, in other words, to perform "weight tuning" for higher-level representation acquisition. We propose two models, Weighted Neural Network (WNN) and Binary-Expectation Neural Network (BENN), which automatically control how much one specific unit contributes to the higher-level representation. The proposed model can be viewed as incorporating a more powerful compositional function for embedding acquisition in recursive neural networks. Experimental results demonstrate the significant improvement over standard neural models.

1 Introduction

Recursive neural network models [1] constitute one type of neural structure for obtaining higher-level representations beyond word-level such as phrases or sentences. It works in a bottom-up fashion on tree structures (e.g., parse trees) in which long-term dependency can be to some extent captured. Figure 1 gives a brief illustration about how recursive neural models work to obtain the distributed representation for the short sentence "*The movie is wonderful*". Suppose h_{is} and $h_{\text{wonderful}}$ are the embeddings for tokens *is* and *wonderful*. The representation for their parent node VP at second layer is given by:

$$h_{VP} = f(W \cdot [h_{is}, h_{wonderful}] + b) \tag{1}$$

where W and b denote parameters involved in the convolutional function. $f(\cdot)$ is the activation function, usually $tanh$ or $sigmod$ or the rectifier linear function.

For NLP tasks, the obtained embeddings could be further fed into task-specific machine learning models[1], through which parameters are to be optimized. Take sentiment analysis as an example, we could feed the aforementioned sentence embedding into a logistic regression model to classify it as either positive or negative. Embeddings are sometimes more capable of capturing latent semantic meanings or syntactic rules within the text than manually developed features, from which many NLP tasks would benefit (e.g., [2, 3]).

Such a type of structure suffers some sorts of intrinsic drawbacks. Revisit Figure 1, common sense tells us that tokens like "*the*", "*movie*" and "*is*" do not contribute much to the sentiment decision but word "*wonderful*" is the key part (and a good machine learning model should have the ability of learning these rules). Unfortunately, the intrinsic structure of recursive neural networks makes it less flexible to get rid of the influence from less sentiment-related tokens. If the keyword "wonderful" hides too deep in the parse tree, for example, as in the sentence "*I studied Russia in Moscow, where*

[1] Of course, embeddings could also be optimized through the task-specific objective functions.

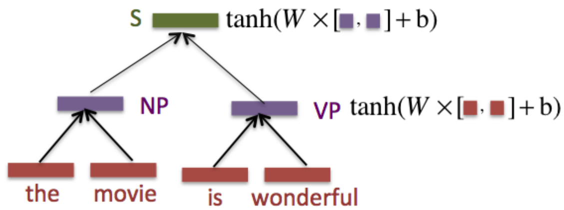

Figure 1: Illustration of Standard Recursive Neural Network for Sentence-level Representation Calculation.

Model	Accuracy
unigram SVM	0.743
Recursive Neural Net	0.730

Table 1: A brief comparison between SVM and standard neural network models for sentence-level sentiment classification using date set from [4]. Neural network models are trained with L2 regularization, using AdaGrad [5] with minibatches (for details about implementations of recursive networks, please see Section 2). Parameters are trained based on 5-fold cross validation on the training data. We report the best performance searching optimum regularization parameter, optimum batch size for mini-batches and convolutional function. Word embeddings are borrowed from Glove [6] with dimensionality of 300, which generates better performance than word2vect, SENNA [7] and RNNLM [8].

all my family think the winter is wonderful", it will takes quite a few convolution steps before the keyword 'wonderful" comes up to the surface, with the consequence that its influence on the final sentence representation could be very trivial. Such an issue, usually referred to as gradient vanishing [9]. is not specific for recursive models, but for most deep learning architectures.

When we compare neural models with SVM, one notable weakness of big-of-word based SVM is its inability of considering how words are combined to form meanings (or order information in other words) [10]. But interestingly, such downside of SVM comes with the advantage of resilience in feature managing as the optimization is "flat-expanded". Low weights will be assigned to less-informative evidence, which could further be pushed to zero by regularization. Table 1 gives a brief comparison between unigram based SVM and neural network models for sentence-level sentiment prediction on Pang et al.'s dataset [4], and as can be seen, in this task, standard neural network models underperform SVM[2].

Revisit the form of Equ.1, there are two straws we can grasp at to deal with the aforementioned problem: (1) expecting the learned feature embeddings for less useful words such as the[3] exert very little influence (for example, a zero vector for the best) (2) expecting the compositional parameters W and b are extremely powerful. For the former, it is sometimes hard, as mostly we borrow (or initialize) word embeddings from those trained from large corpus (e.g., word2vec, RNNLM [8, 11], SENNA [7]), rather than training embeddings from task-specific objective functions as neural models can be easily over fitted given the small amount of training data[4].

Regarding the latter issue, several alternative compositional functions have been proposed to enable more varieties in composition to cater. Recent proposed approaches include, for example, Matrix-Vector RNN [12], which represents every word as both a vector and a matrix, RNTN [2] which allows greater interactions between the input vectors, and the algorithm presented in [13] which

[2]To note, results here are not comparable with Socher et al.'s work [2] which obtains state-of-art performance in sentiment classification, as here labels at sentence-level constitute only sort of supervision for both SVM and neural network models (for details, see footnote 7).

[3]We just use this example for illustration. Practically, the might be a good sentiment indicator as it usually co-appears with superlatives.

[4]There are cases, for example, [2], where task-specific word embeddings are learned. But it requires sufficient training data to avoid over fitting. For example, Socher et al.'s work labels every single node as positive/negative/neutral along parse trees (with a total number of more than 200,000 phrases).

associates different labels (e.g., POS tags, relation tags) with different sets of compositional parameters. These approaches to some extent enlarge the power of compositional functions.

In this paper, we borrow the idea of "weight tuning" from feature based SVM and try to incorporate such idea into neural architectures. To achieve this goal, we propose two recursive neural architectures, Weighted Neural Network (WNN) and Binary-Expectation Neural Network (BENN). The major idea involved in the proposed approaches is to associate each node in the recursive network with additional parameters, indicating how important it is for final decision. For example, we would expect such type of a structure would dilute the influence of tokens like "the" and "movie" but magnifies the impact of tokens like "wonderful" and "great" in sentiment analysis tasks. Parameters associated with proposed models are automatically optimized through the objective function manifested by the data. The proposed model combines the capability of neural models to capture the local compositional meanings with weight tuning approach to reduce the influence of undesirable information at the same time, and yield better performances in a range of different NLP tasks when compared with standard neural models.

The rest of this paper is organized as follows: Section 2 briefly describes the related work. The details of WNN and BENN are illustrated in Section 4 and experimental results are presented in Section 5, followed by a brief conclusion.

2 Related Work

Distributed representations, calculated based on neural frameworks, are extended beyond token-level, to represent N-grams [14], phrases [2], sentences (e.g., [3, 15]), discourse [16, 13], paragraphs [17] or documents [18]. Recursive and recurrent [19, 20] models constitute two types of commonly used frameworks for sentence-level embedding acquisition. Different variations of recurrent/recursive models are proposed to cater for different scenarios (e.g., [3, 2]). Other recently proposed approaches included sentence compositional approach proposed in [21], or paragraph/sentence vector [17] where representations are optimized through predicting words within the sentence.

Neural network architecture sometimes requires a vector representation of each input token. Various deep learning architectures have been explored to learn these embeddings in an unsupervised manner from a large corpus [22, 23, 24, 25], which might have different generalization capabilities and are able to capture the semantic meanings depending on the specific task at hand.

Both of the proposed architectures are in this work inspired by the long short-term memory (LSTM) model, first proposed by Hochreiter and Schmidhuber back in 1990s [26, 27] to process time sequence data where there are very long time lags of unknown size between important events[5]. LSTM associates each time with a series of "gates" to determine whether the information from early time-sequence should be forgotten [27] and when current information should be allowed to flow into or out of the memory. LSTM could partially address gradient vanishing problem in recurrent neural models and have been widely used in machine translation [28, 29]

3 "Weight Tuning" for Neural Network

Let s denote a sequence of token $s = \{w_1, w_2, ..., w_{n_s}\}$. It could be phrases, sentences etc. Each word w is associated with a specific vector embedding $\mathbf{e_w} = \{e_w^1, e_w^2, ..., e_w^K\}$, where K denotes the dimensionality of the word embedding. We wish to compute the vector representation for sentence s, denoted as $h_s = \{h_s^1, h_s^2, ..., h_s^K\}$ based on parse trees using recursive neural models. Parse tree for each sentence is obtained from Stanford Parser [30].

3.1 WNN for Recursive Neural Network

For any node C in the parse tree, it is associated with representation h_C. The basic idea of WNN is to associate each node C with an additional weight variable M_C, which is in range (0,1), to denote the importance of current node. Technically, M_C is used to pushing the output representation of not-useful node towards the direction of 0 and retain relatively important information.

[5] http://en.wikipedia.org/wiki/Long_short_term_memory

We expect that information regarding the importance of current node (e.g., whether it is relevant to positive/negative sentiment) is embedded in its representation h_C. So we use a convolution function to enable this type of information to emerge to the surface from the following compositional functions:

$$R_C = f(W_M \cdot h_C + b_M) \tag{2}$$

$$M_C = sigmod(U_M^T \cdot R_C) \tag{3}$$

where W_M is a $D \times K$ dimensional matrix and b_B is the $1 \times K$ bias vector. R_C is a K dimensional intermediate vector. Such implementation can be viewed as using a three-layer neural model with D latent neurons for an output projected to a [0,1] space.

Let $output(C)$ denote the output from node C to its parent. In WNN, $output(C)$ would consider both current information, which is embedded in the embedding h_C and its related importance M_C. $output(C)$ is therefore given by

$$output(C) = M_C * h_C \tag{4}$$

Recall the example in Figure 1, we have:

$$\begin{aligned} output(the) &= M_{the} \cdot h_{the} \\ output(movie) &= M_{movie} \cdot h_{movie} \end{aligned} \tag{5}$$

If the model thinks not too much relevant information embedded in h_C, the value of M_C would be small, pushing the output vector towards 0. The representations for parents, for example VP and NP in Figure 1, are therefore computed as follows:

$$\begin{aligned} h_{VP} &= tanh(W_B \cdot [output(is), output(wonderful)]) \\ h_{NP} &= tanh(W_B \cdot [output(the), output(movie)]) \end{aligned} \tag{6}$$

where W_B denotes a $K \times 2K$ dimensional matrix and $[output(is), output(wonderful)]$ denotes the concatenation of the two vectors. In an optimum situation, M_{the} and M_{movie} will take the values around 0, leading to the representation of node NP to an around-zero vector.

Training WNN For illustration purpose, we use a binary classification task to show how to train WNN. To note, the described training approach applies to other situations (e.g., multi-class classification, regression) with minor adjustments.

In a binary classification task, each sequence is associated with a gold-standard label y_s. y_s takes value of 1 if positive and 0 otherwise. Standardly, to determine the value of y_s, we feed the representation h_s into a logistic regression model:

$$p(y_s = 1) = sigmod(U^T h_s + b) \tag{7}$$

where U^T is a $1 * K$ vector and b denotes the bias. Then by adding the regularization part parameterized by Q, the loss function $J(\Theta)$ for the training dataset is given by:

$$J(\Theta) = -\log[p(y_s = 1)^{y_s} \cdot (1 - p(y_s = 1))^{1-y_s}] + Q \sum_{\theta \in \Theta} \theta^2 \tag{8}$$

Revisit the example in Figure 1, for any parameter θ to optimize, the calculation for gradient $\partial J / \partial \theta$ is trivial once $\partial [M_{VP} \cdot h_{VP}]/\partial \theta$ and $\partial [M_{NP} \cdot h_{NP}]/\partial \theta$ are obtained, which are given by:

$$\frac{\partial M_{VP} \cdot h_{VP}}{\partial \theta} = M_{VP} \frac{\partial h_{VP}}{\partial \theta} + \frac{\partial M_{VP}}{\partial \theta} h_{VP} \tag{9}$$

To note, h_{VP} is embraced in M_{VP}. As all components in Equ 9 are continuous, the gradient can be efficiently obtained from standard backpropagation [31, 32].

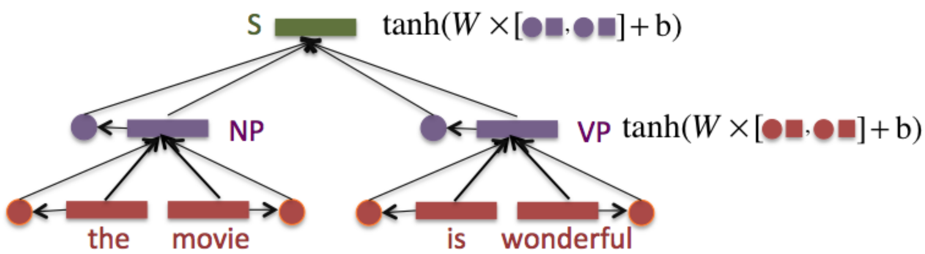

Figure 2: Illustration of WNN.

3.2 BENN for Recursive Neural Network

BENN associates each node with a binary variable B_C, which is sampled from a binary distribution parameterized by L_C. L_C is a scalar fixed to the range of [0,1], indicating the possibility that current node should pass information to its ancestors. L_C is obtained in the similar ways as in WNN by using a convolution to project the current representation h_C to a scalar lying within [0,1].

$$R_C = f(W_B \cdot h_C + b_B) \tag{10}$$

$$L_C = \mathrm{sigmod}(U_B^T \cdot R_C) \tag{11}$$

$$B_C \sim \mathrm{binary}(L_C) \tag{12}$$

For smoothing purpose, in BENN, current node C outputs the expectation of embedding h_C to its parent, as given by:

$$\mathrm{output}(C) = E[h_C] \tag{13}$$

Take the case in Figure 1 as an example again, vector h_{NP} will therefore follow the following distribution:

$$\begin{aligned} &p(h_{NP} = \tanh(W_B[h_{the}, h_{movie}])) = L_{the} \cdot L_{movie} \\ &p(h_{NP} = \tanh(W_B[0, h_{movie}])) = (1 - L_{the}) \cdot L_{movie} \\ &p(h_{NP} = \tanh(W_B[h_{the}, 0])) = L_{the} \cdot (1 - L_{movie}) \\ &p(h_{NP} = \tanh(W_B[0, 0])) = (1 - L_{the}) \cdot (1 - L_{movie}) \end{aligned} \tag{14}$$

$E[h_{NP}]$ can be further obtained based on such distribution

$$E[h_{NP}] = \sum_h P(h_{NP} = h) \cdot h \tag{15}$$

To note, for leaf nodes, $E[h_C] = h_C$.

Training BENN For training, we again use binary sentiment classification for illustration. For any sentence s with label y_s, we have

$$p(y_s = 1) = \mathrm{sigmod}(U^T E[h_s] + b) \tag{16}$$

With respect to any given parameter θ, the derivative of $E[h_s]$ is further given by:

$$\begin{aligned} \frac{\partial E(h_s)}{\partial \theta} &= \frac{\partial L_{NP} \cdot L_{VP} \cdot \tanh(W_B[E(h_{NP}), E(h_{VP})])}{\partial \theta} \\ &+ \frac{\partial (1 - L_{NP}) \cdot L_{VP} \cdot \tanh(W_B[0, E(h_{VP})])}{\partial \theta} \\ &+ \frac{\partial L_{NP} \cdot (1 - L_{VP}) \cdot \tanh(W_B[E(h_{NP}), 0]}{\partial \theta} \\ &+ \frac{\partial (1 - L_{NP}) \cdot (1 - L_{VP}) \cdot \tanh(W_B[0, 0])}{\partial \theta} \end{aligned} \tag{17}$$

With all components being continuous, the gradient can be efficiently obtained from standard back-propagation.

Model		Accuracy
Recursive	Standard (GLOVE)	0.730
	Standard (learned)	0.658
	MV-RNN	0.704
	RNTN	0.760
	Label-Specific	0.768
	WNN	0.778
	BENN	0.772
SVM	unigram	0.743

Table 2: Binary Sentiment Classification with Supervision only at Sentence Level. Word embeddings are initialized from 300 dimensional embeddings borrowed from GLOVE [6].

4 Experiment

We perform experiments to better understand the behavior of the proposed models compared with standard neural models (and other variations). To achieve this, we implement our model on problems that require fixed-length vector representations for phrases or sentences.

4.1 Sentiment Analysis

Sentence-level Labels We first perform experiments on dateset from [4]. In this setting, binary labels at the top of sentence constitute the only resource of supervision (to note, it is different from settings described in [2]). All neural models adopt the same settings for fair comparison: L2 regularization, gradient decent based on AdaGrad with mini batch size of 25, tuned parameters for regularization on 5-fold cross validation.

For standard neural models, we implement two settings: standard (GLOVE) where word embeddings are directly fixed to GLOVE and standard (learned) where word embeddings are treated as parameters to optimize in the framework. Additionally, we implemented some recent popular variations of recursive models with more sophisticatedly designed compositional functions, including:

- MV-RNN (Matrix-Vector RNN): which was proposed in [12] which represents every node in a parse tree as both a vector and a matrix. Given the vector representation h_{C_1}, matrix representation V_{C_1} for child node C_1, h_{C_2} and V_{C_2} for child node C_2, the vector representation h_p and matrix representation V_p for parent p are given by:

$$\begin{aligned} h_p &= f(W_1[V_{C_1} \cdot h_{C_2}, V_{C_2} \cdot h_{C_1}]) \\ V_p &= f(W_1[V_{C_1}, V_{C_2}]) \end{aligned} \tag{18}$$

  We fix word vector embeddings using SENNA and treat matrix representations as parameters to optimize.

- RNTN (Recursive Neural Tensor Network): proposed in [2]. Given h_{C_1} and h_{C_2} for children nodes, RNTN computes parent vector h_p in the following way:

$$h_p = f([h_{C_1}, h_{C_2}]^T V[h_{C_1}, h_{C_2}] + W[h_{C_1}, h_{C_2}]) \tag{19}$$

- Label-specific: associate each of the sentence roles (i.e., VP, NP or NN) with a specific composition matrix.

We report results in Table 2. As discussed earlier, standard neural models underperform the bag of word models. To note, for derivations of standard neural models such as Standard (learned) and MV-RNN with many more parameters to learn, the performance is even worse due to over-fitting. WNN and BENN , although not significantly output bag of words SVM, generates better results, yielding significant improvement over standard neural models and existing revised versions. Figure 3 illustrates the automatic learned muted factor M_C regarding different nodes in the parse tree based on recursive network. As we can observe , the model is capable of learning the proper weight of vocabularies, assigning larger weight values to important sentiment indicators (e.g., wonderful, silly and tedious) and suppressing the influence of less important ones. We attribute the better performance of proposed models over standard neural models to such automatic weight-tuning ability.

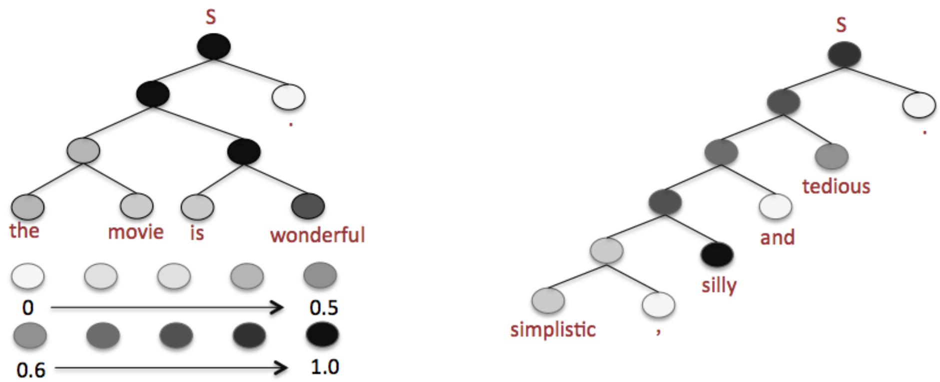

Figure 3: Visual illustration of automatic learning of weight M_C associated with each node in WMM.

To note, in this scenario, we are not claiming that we generate state-of-art results using the proposed model. More sophisticated bag-of-word models, for example, (e.g., [33]) can generate better performance that what the proposed models achieve. The point we wish to illustrate here is that the proposed models provide a promising perspective over standard neural models due to the "weight tuning" property. And in the cases where more detailed data is available to capture the compositionally, the proposed models hold promise to generate more compelling results, as we will illustrate in Socher et al's setting for sentiment analysis.

Socher et al's settings We now consider Socher et al's dataset [2] for sentiment analysis, where contains gold-standard labels at every phrase node in the parse tree. The task could be considered either as a 5-way fine-grained classification task where the labels are very-negative/negative/neutral/positive/very-positive or a 2-way coarse-way as positive/negative based on labeled dataset. We follow the experimental protocols described in [2] (word embeddings are treated as parameters to learn rather than fixed to externally borrowed embeddings). In this work we only consider labeling the full sentences.

In addition to varieties of neural models mentioned in Socher et al's work, we also report the performance of recently proposed *paragraph vector model* [17], which first obtains sentence embeddings in an unsupervised manner by predicting words within the context and then feeds the pre-obtained embeddings into a logistic regression model. *paragraph vector* achieves current the state-of-art performance regarding Socher et al's dataset.

Performances are reported in Table 3. As can be seen, the proposed approach slightly underperforms current state-of-art performance achieved by *paragraph vector* but outperforms all the other versions of recursive neural models, indicating the adding "weight tuning" parameters indeed leads to better compositionally.

To note, when there is more comprehensive dataset which we can rely on to obtain the favorable task-specific word embeddings, compositionally plays an important role in deciding whether the review is positive or negative by harnessing local word order information. In that case, neural models exhibit its power in capturing local evidence from the composition, leading to significantly better performance than all bag-of-words based models (i.e., SVM and Bigram Naives Bayes).

4.2 Document-level Sentiment Analysis on IMDB dataset

We move on to sentiment analysis at document level. We use the IMDB dataset proposed by Maas et al. [33]. The dataset consists of 100,000 movie reviews taken from IMDB and each movie review contains several sentences. We follow the experimental protocols described in [33].

We first train word vectors from word2vect using the 75,000 training documents. Next we train the compositional functions using the 25,000 labeled documents by keeping the word embedding fixed. We first obtain sentence-level representations using WNN/BENN (recursive). As each review contains multiple sentences, we convolute sentence representations to one single vector using

Model	Fine-grained	Coarse-grained
SVM	0.407	0.794
Bigram Naives	0.419	0.831
Recursive	0.432	0.824
MV-RNN	0.444	0.829
RNTN	0.457	0.854
Paragraph Vector	0.487	0.878
WNN	0.482	0.865
BENN	0.475	0.870

Table 3: The performance of proposed approaches compared with other methods on Stanford sentiment treebank dataset. Baseline performances are reported from [2, 17].

Model	Precision
SVM-unigram	0.869
SVM-bigram	0.892
recursive+recurrent	0.870
WNN	0.902
BENN	0.910

Table 4: The performance of proposed model compared to other approaches on binary classification on IMDB dataset. Results for baselines are reported from [33]. To note, the reported results here underperform current state-of-the-art performances. Paragraph vectors [17] reported an accuracy of 92.58 in terms of IMDB dataset.

WNN/BENN recurrent network. We cross validate parameters using the labeled documents and test the models on the 25,000 testing reviews.

The results of our approach and other baselines are reported in Table 5. As can be seen, for long documents, bag-of-words (both unigram and diagram) perform quite well and it is difficult to beat. Standard neural models again do not generate competent results compared with bag of word models in this task. But by incorporating weighted tuning mechanism, we got much better performance, roughly 5% when compared against standard neural models. Although WNN and BENN still underperform current state-of-art model Paragraph Vector [17], they produces better performance than bag-of-word models.

4.3 Sentence Representations for Coherence Evaluation

Sentiment analysis forces more on the semantic perspective of meaning. Next we turn to a more syntactic oriented task, where we obtain sentence-level representations based on the proposed model to decide the coherence of a given sequence of sentences.

We use corpora widely employed for coherence prediction [34, 35]. One contains reports on airplane accidents from the National Transportation Safety Board and the other contains reports about earthquakes from the Associated Press. Standardly, we use pairs of articles, one containing the original document order which is assumed to be coherent and used as positive examples, and the other a random permutation of the sentences from the same document, which are treated as not-coherent examples. We follow the protocols introduced in [34, 36, 37] by considering a window approach and feeding the concatenation of representations of adjacent sentences into a logistic regression model, to be classified as either coherent or non-coherent. In test time, we assume that the model makes a right decision if the original document gets a score higher than the one with random permutations. Current state-of-art performance regarding this task is obtained by using standard recursive network as described in [37].

Table 5 illustrates the performance of different models. Entity-grid model [34] generates state-of-art performance among all non-neural network models. As can be seen, neural models perform pretty well in this task when compared against existing feature based algorithm. From the reported results, better sentence representations are obtained by incorporating “weighted tuning” properties, pushing the state of art of this task to the accuracy of 0.936.

Model	Accuracy
WNN-recursive	0.930
BENN-recursive	0.936
recursive	0.920
Entity-Grid	0.888

Table 5: Comparison of Different Coherence models. Reported baseline results are reprinted from [34].

5 Conclusion

In this paper, we propose two revised versions of neural models, WNN and BENN for obtaining higher level feature representations for a sequence of tokens. The proposed framework automatically incorporates the concept of "weight tuning" of SVM into the DL architectures which lead to better higher-level representations and generate better performance against standard neural models in multiple tasks. While it still underperforms bag-of-word models in some cases, and the newly proposed paragraph vector approach, it provides as an alternative to existing recursive neural models for representation learning.

To note, while we limit our attentions to recursive models in this work, the idea of weight tuning in WNN and BENN, that associates nodes in neural models with additional weighed variables is a general one and can be extended to many other deep learning models with minor adjustment.

References

[1] Ronald J Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280, 1989.

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

[3] Ozan Irsoy and Claire Cardie. Bidirectional recursive neural networks for token-level labeling with structure. *arXiv preprint arXiv:1312.0493*, 2013.

[4] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics, 2002.

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

[6] RichardSocher JeffreyPennington and ChristopherD Manning. Glove: Global vectors for word representation. 2014.

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

[8] Tomas Mikolov, Stefan Kombrink, Anoop Deoras, Lukar Burget, and J Cernocky. Rnnlm-recurrent neural network language modeling toolkit. In *Proc. of the 2011 ASRU Workshop*, pages 196–201, 2011.

[9] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *Neural Networks, IEEE Transactions on*, 5(2):157–166, 1994.

[10] Raymond J Mooney and Razvan C Bunescu. Subsequence kernels for relation extraction. In *Advances in neural information processing systems*, pages 171–178, 2005.

[11] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *INTERSPEECH*, pages 1045–1048, 2010.

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

[13] Jiwei Li, Rumeng Li, and Eduard Hovy. Recursive deep models for discourse parsing. 2014.

[14] Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 151–161. Association for Computational Linguistics, 2011.

[15] Phil Blunsom, Edward Grefenstette, Nal Kalchbrenner, et al. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, 2014.

[16] Yangfeng Ji and Jacob Eisenstein. Representation learning for text-level discourse parsing. 2014.

[17] Quoc V Le and Tomas Mikolov. Distributed representations of sentences and documents. *arXiv preprint arXiv:1405.4053*, 2014.

[18] Misha Denil, Alban Demiraj, Nal Kalchbrenner, Phil Blunsom, and Nando de Freitas. Modelling, visualising and summarising documents with a single convolutional neural network. *arXiv preprint arXiv:1406.3830*, 2014.

[19] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *Signal Processing, IEEE Transactions on*, 45(11):2673–2681, 1997.

[20] Ilya Sutskever, James Martens, and Geoffrey E Hinton. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1017–1024, 2011.

[21] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, June 2014.

[22] Yoshua Bengio, Holger Schwenk, Jean-Sébastien Senécal, Fréderic Morin, and Jean-Luc Gauvain. Neural probabilistic language models. In *Innovations in Machine Learning*, pages 137–186. Springer, 2006.

[23] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008.

[24] Andriy Mnih and Geoffrey Hinton. Three new graphical models for statistical language modelling. In *Proceedings of the 24th international conference on Machine learning*, pages 641–648. ACM, 2007.

[25] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[26] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[27] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. *Neural computation*, 12(10):2451–2471, 2000.

[28] Martin Sundermeyer, Tamer Alkhouli, Joern Wuebker, and Hermann Ney. Translation modeling with bidirectional recurrent neural networks. In *Proceedings of the Conference on Empirical Methods on Natural Language Processing, October*, 2014.

[29] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[30] Richard Socher, John Bauer, Christopher D Manning, and Andrew Y Ng. Parsing with compositional vector grammars. In *In Proceedings of the ACL conference*. Citeseer, 2013.

[31] Christoph Goller and Andreas Kuchler. Learning task-dependent distributed representations by backpropagation through structure. In *Neural Networks, 1996., IEEE International Conference on*, volume 1, pages 347–352. IEEE, 1996.

[32] Richard Socher, Christopher D Manning, and Andrew Y Ng. Learning continuous phrase representations and syntactic parsing with recursive neural networks. In *Proceedings of the NIPS-2010 Deep Learning and Unsupervised Feature Learning Workshop*, pages 1–9, 2010.

[33] Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 142–150. Association for Computational Linguistics, 2011.

[34] Regina Barzilay and Lillian Lee. Catching the drift: Probabilistic content models, with applications to generation and summarization. In *HLT-NAACL*, pages 113–120, 2004.

[35] Regina Barzilay and Mirella Lapata. Modeling local coherence: An entity-based approach. *Computational Linguistics*, 34(1):1–34, 2008.

[36] Annie Louis and Ani Nenkova. A coherence model based on syntactic patterns. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1157–1168. Association for Computational Linguistics, 2012.

[37] Jiwei Li and Eduard Hovy. A model of coherence based on distributed sentence representation.