Investigation of Recurrent Neural Networks in the field of Sentiment Analysis

Kiran Baktha, and B K Tripathy, *Senior Member*, *IEEE*

***Abstract*—Recurrent Neural Networks(RNNs) are popular deep learning architectures used in Natural Language Processing for analyzing sentiments in sentences. The recurrent nature of these networks enable them to use information from previous time steps. In this paper, we analyze the performance of three RNNs namely vanilla RNNs, Long Short-Term Memory(LSTM) and Gated Recurrent Units(GRU). Both unidirectional and bidirectional nature of these networks are considered. Pre-trained word vectors from the Google News dataset are used. We evaluate the performance of these networks on the Amazon health product reviews dataset and sentiment analysis benchmark datasets SST-1 and SST-2.**

***Index Terms*—Deep Learning, Natural Language Processing, Recurrent Neural Networks, Sentiment Analysis**

I. Introduction

Sentiment Analysis also known as opinion mining is the process of determining the emotions behind words, sentences or documents which is very useful in analyzing public opinions. Convolutional and Recurrent Neural Networks are widely applied for classification of texts. The main advantage of RNNs is that they can be used to model temporal sequences with variable length, which provides added flexibility in analyzing reviews of different lengths. RNNs have proved their superiority in many NLP tasks including learning language models that can outperform n-grams [1-4]. Various other applications of RNN include those involving sequential data such as speech processing, generating text and image processing. Several architectures involving RNNs have been proposed for sentiment analysis[5-8]. Timmaraju *et al*. [10], proposed a recursive-recurrent neural network which can be used for analyzing sentiment on movie reviews. Recursive neural networks are similar in spirit to recurrent neural networks but use tree topology for its time steps as opposed to chain topology in recurrent neural networks. Bidirectional recurrent neural networks take previous and future information into consideration. Zhou *et al*. [12] proposed a model by using two-dimensional max pooling on bidirectional LSTMs which had achieved impressive accuracy of 52.4% and 89.5% on SST-1 and SST-2 respectively. Several other architectures combining convolutional and recurrent neural network architectures have been proposed [7,11].

Even though various complex neural network architectures have been proposed, there is no sole literature dedicated to providing an in-depth analysis on the standard types on recurrent neural networks for sentiment analysis. Motivated by this drawback, our main objective is to experiment on the three types of RNNs in predicting the sentiment of reviews. We use deep RNNs and limit our analysis to three hidden recurrent layers.

We organize the paper as follows: Section II explains the background behind the structures of RNNs and their equations, Section III contains detailed information on the experiments conducted including the hyperparameters chosen for training, Section IV contains the results obtained and Section V concludes the paper.

II. Background

A. Vanilla RNN

Recurrent neural networks are artificial neural networks with directed cycles. At each time-step, the current input and the previous time step's hidden state are fed as inputs after passing though their weight matrices U and W respectively as shown in Fig. 1.

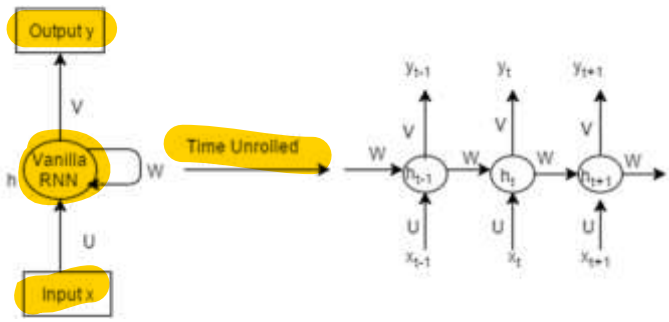

Fig. 1. A Vanilla RNN structure and its time unrolled version used for forward computation

Kiran Baktha, is with the School of Electronics and Communication Engineering, Vellore Institute of Technology, Vellore, India. (e-mail: sundarambaktha@hotmail.com)

B. K. Tripathy is a faculty member in the School of Computer Science and Engineering, Vellore Institute of technology, Vellore, India. (e-mail: tripathybk@vit.ac.in

978-1-5090-3800-8/17/$31.00 ©2017 IEEE

Equations involved in vanilla RNN are:

$$h_t = f_h(U.x_t + W.h_{t-1} + b) \quad (1)$$

$$y_t = f_o(V.\mathrm{h}_t) \quad (2)$$

Where (1) is to calculate the hidden state value h_t and (2) to calculate the output y_t. W is the recurrent weight matrix, U is the input to hidden layer matrix and V is the hidden layer to output matrix. Vanilla RNNs are difficult to train due to the exploding and vanishing gradient problems [5] which led to the development of LSTMs and GRUs.

B. *Long Short-Term Memory(LSTM)*

LSTMs were introduced by Hochreiter *et al*. in [2] which is one of the most commonly used RNN. By using a variety of gates, these networks can regulate the propagation of activations along the network which enables it to learn when to ignore a current input, when to remember the past hidden state or when to emit a non-zero input. These networks are efficient at remembering information for long or short durations of time and hence the name Long Short-Term Memory. The structure of LSTM is given in Fig. 2.

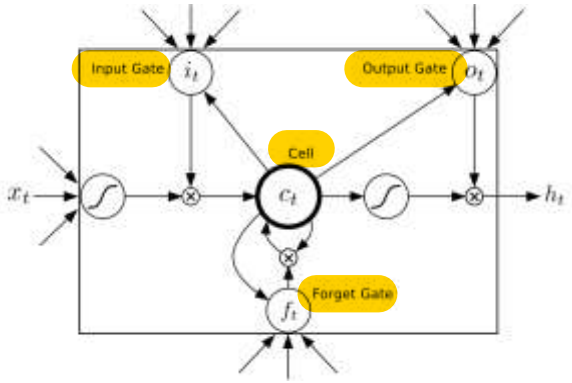

Fig. 2. LSTM block diagram [13]

There are three gates used: input gate determines how much the current input is let through, forget gate determines how much of the previous state is let through and output gate determines how much the current node influences the external network. The equations involved in computing the values of the LSTM are:

$$i_t = \mathrm{logistic}(W_i x_t + U_i h_{t-1} + b_i) \quad (3)$$

$$f_t = \mathrm{logistic}(W_f x_t + U_f h_{t-1} + b_f) \quad (4)$$

$$o_t = \mathrm{logistic}(W_o x_t + U_o h_{t-1} + b_o) \quad (5)$$

$$\tilde{c}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c) \quad (6)$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t \quad (7)$$

$$h_t = o_t \circ \tanh(c_t) \quad (8)$$

Equations (3), (4) and (5) are used to calculate the value of the input, forget and output gates respectively. Equation (6) is used to calculate the value of the proposed state $\tilde{c}_t$ which is substituted to find the current state c_t in (7). Finally, the hidden state h_t is calculated using (8). Each of the three gates and the proposed state have their own weight matrices U and W with a bias vector b that is learnt during training. '$\circ$' indicates element-wise multiplication.

C. *Gated Recurrent Unit (GRU)*

Another variant of the RNN is the Gated Recurrent Unit introduced by Cho, *et al*. in [1]. A GRU is similar to an LSTM but has only two gates: reset gate which determines how the new input and previous information are combined and the update gate which determines how much previous information to pass. A basic block diagram of GRU is given in Fig. 3.

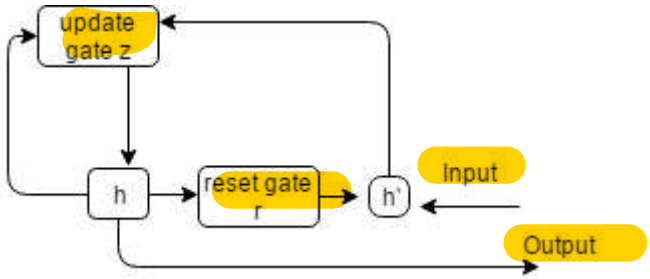

Fig. 3. GRU block diagram

Equations governing GRU are given by:

$$z_t = \mathrm{logistic}(U_z \mathrm{x}_t + \mathrm{W}_z \mathrm{h}_{t-1}) \quad (9)$$

$$r_t = \mathrm{logistic}(U_r \mathrm{x}_t + \mathrm{W}_r \mathrm{h}_{t-1}) \quad (10)$$

$$h' = \tanh(U_h x_t + W_h(h_{t-1} \circ r)) \quad (11)$$

$$h_t = (1-z) \circ h_{t-1} + z_t \circ h'_t \quad (12)$$

Equations (9) and (10) are used to compute the update and reset gates respectively. h' is the candidate hidden activation function defined in (11) which is used to compute the hidden activation h given by (12).

D. *RMSProp Optimizer*

In our experiments, we use the RMSprop optimizer which was proposed by Geoffrey Hinton in his Coursera course titled "Neural Networks for Machine Learning". It is the most commonly used optimizer for training recurrent neural networks and is a version of Resilient Propagation (rprop) for mini-batch learning. RMSProp divides a gradient by a running average of its recent magnitude. We first calculate the running average r_t given by (13).

$$r_t = (1-\gamma) f'(\theta_t)^2 + \gamma r_{t-1} \quad (13)$$

γ is the decay term and $f'(\theta_t)$ is the derivative of the loss

function with respect to the parameters θ_t at time step t. We then divide the learning rate by the root mean square of the running average and the update rule is given by the equation:

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{r_t + \varepsilon}} f'(\theta_t) \quad (14)$$

where α is the learning rate and ε is the fuzz factor.

E. Bidirectional RNNs

These RNNs were proposed by Schuster and Paliwal in [6]. The basic structure is given in Fig. 4. where there are two layers (see Fig. 4) between the input and the output. The first layer is the traditional unidirectional RNN and the second layer processes the input sequence backwards. This way both past and future information are available at every time step.

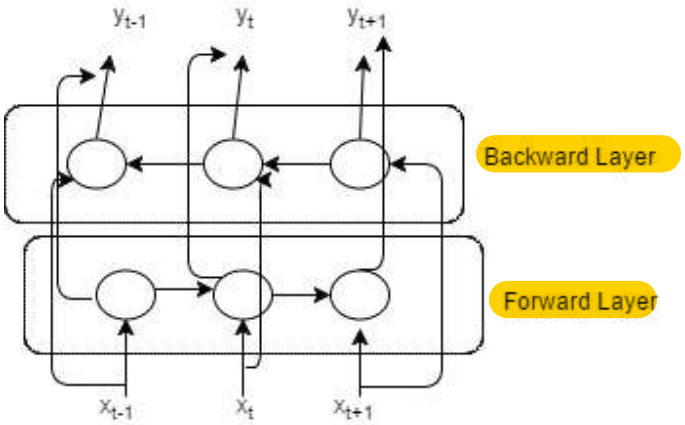

Fig. 4. Bidirectional RNN structure

While training a BNN we need to learn both forward and backward weights which increases the number of parameters of the network. Forward Layer computation if given by (15) and backward layer is computed using (16). Both the hidden layer values at a particular time step are combined in (17) to determine the output.

$$\vec{h}_t = f_h(\vec{U}.x_t + \vec{W}.\vec{h}_{t-1} + \vec{b}) \quad (15)$$

$$\overleftarrow{h}_t = f_h(\overleftarrow{U}.x_t + \overleftarrow{W}.\overleftarrow{h}_{t-1} + \overleftarrow{b}) \quad (16)$$

$$y_t = f_o(\vec{V}\ .\ \vec{h}_t + \overleftarrow{V}\ .\ \overleftarrow{h}_t) \quad (17)$$

III. Methodology

A. Word Embeddings

We feed pre-trained word vectors as inputs to the recurrent neural networks. The word vectors are from Google's word2vec model pre-trained on Google news dataset of about 100 billion words. The model outputs 300 dimensional vectors for 3 million words and phrases and uses continuous bag of words architecture. We tried to use custom trained word vectors on the vocabulary of each dataset. On exploring these custom word vectors, we found that the learnt vectors from the Amazon dataset were very efficient but the SST model could not output impressive word vectors. For this reason, we decided to use Google's pre-trained word embeddings.

B. Model

Basic layout of the model is given in Fig. 5. The input which are reviews is fed to the embedding layer which converts each word to a 300-dimensional vector. This vector is fed to the RNN structure with 1,2 or 3 hidden layers. The output from the RNN is fed to a dense layer which predicts the output.

Fig. 5. Model Layout

The dense layer has 1 node for a binary sentiment classification and 5 nodes for multiclass sentiment problem.

C. Hyperparameters

Each hidden layer in the recurrent neural network has 300 nodes which is the dimension of the word vector. The RMSprop parameters as set to their default values with learning rate equal to 0.001 and γ as 0.9. The network is trained with mini-batch gradient descent with a batch size of 64. The dense layer has sigmoid activation for binary sentiment classification and softmax activation function for multi-class sentiment classification. We have used dropouts [9] to prevent overfitting. Embedding layer uses a dropout of 0.2 while the recurrent structures have a dropout of 0.4.

D. Datasets Used

- **Amazon Product Reviews** – We took the Health and Personal Care product reviews from Amazon website which were accessible in University of California, San Diego website through Julian McAuley [3]. We preprocessed the dataset to include only 10,000 reviews such that 50% positive and negative reviews are included making it a binary classification problem. 10% of the data is used for testing and 10% for validation purposes.
- **SST-1** – Stanford Sentiment Treebank. The dataset contains movie reviews which are labelled by Socher *et al.* [8]. Reviews are labelled into 5 classes (very positive, positive, neutral, negative and very negative) which makes it a multi-class classification problem. 8544 reviews are used for training, 1101 reviews for validation and 2210 reviews for testing.
- **SST-2** – It is similar to SST-1 but with neutral reviews removed and labels are made binary. Therefore, this dataset becomes a binary classification problem. 6920 reviews are used for training, 872 reviews for validation and 1821 reviews for testing.

IV. Results

The results on the various datasets are summarized as follows:

Each entry in the column is the accuracy percentage obtained on running the network in the specific dataset. The prefix "Bi" indicates Bidirectional structure. The loss function used is categorical cross-entropy for multiclass sentiment classification and binary cross-entropy for binary sentiment classification.

TABLE I
ACCURACY ON THE AMAZON REVIEWS DATASET

RNN Type	1-Layer	2-Layers	3-Layers
VanillaRNN	57.30%	56.00%	51.30%
LSTM	78.10%	66.80%	51.30%
GRU	**83.90%**	77.50%	78.60%
Bi-VanillaRNN	58.00%	58.40%	56.10%
Bi-LSTM	79.20%	67.50%	54.80%
Bi-GRU	**81.10%**	62.70%	59.40%

TABLE II
ACCURACY ON SST-1 DATASET

RNN Type	1-Layer	2-Layers	3-Layers
VanillaRNN	35.29%	26.86%	25.56%
LSTM	43.98%	38.47%	28.83%
GRU	**44.61%**	43.75%	41.16%
Bi-VanillaRNN	37.0%	40.36%	25.11%
Bi-LSTM	42.89%	42.17%	30.0%
Bi-GRU	**43.81%**	42.69%	28.64%

TABLE III
ACCURACY ON SST-2 DATASET

RNN Type	1-Layer	2-Layers	3-Layers
VanillaRNN	75.78%	75.06%	75.67%
LSTM	82.2%	74.46%	64.47%
GRU	**84.40%**	83.36%	82.86%
VanillaRNN	78.96%	78.58%	80.12%
LSTM	**81.54%**	79.51%	63.86%
GRU	81.32%	81.10%	73.06%

TABLE IV
CROSS-ENTROPY LOSS OBTAINED

RNN Type (1-Layer)	Amazon Dataset	SST-1	SST-2
GRU	0.3865	1.2567	0.3769
Bi-GRU	0.4203	1.2814	0.3966

For both unidirectional and bidirectional structures, single layer GRU achieved very high accuracy in all the tree datasets. Three layered RNN structures do not seem to work very well for the datasets. We believe this is because the model becomes over complex leading to lower accuracy. For huge datasets involving lengthy reviews, more layers might improve the performance. Unidirectional structures performed quite better than the bidirectional structures. VanillaRNNs could be considered if computation time is very important because they have the least number of parameters so, learn the fastest. Their accuracy is impressive in SST-1 and SST-2 datasets but have poor performance on the Amazon Product Reviews dataset.

V. CONCLUSIONS

In this paper, we explored standard Recurrent Neural Network structures on three sentiment analysis datasets. We found Gated Recurrent Units to be the best choice in terms of accuracy. There is no need to use deep RNNs unless the dataset is huge with very lengthy reviews. As future scope, hybrid models involving convolutional neural networks or highway networks could be experimented with RNNs. More research can be dedicated towards investigating how complex optimization methods such as the hessian-free optimizer which approximates the Hessian matrix rather than having to compute it and can be used in Vanilla RNNs perform analyzing sentiments.

REFERENCES

[1] Cho, K. et al. "Learning phrase representations using RNN encoder-decoder for statistical machine translation". In *Proc. Conference on Empirical Methods in Natural Language Processing, 2014, pp.1724–1734.*

[2] Hochreiter, S. & Schmidhuber, J. "Long short-term memory*". Neural Computation. Volume 9, Issue 8, 1997, pp. 1735–1780.*

[3] McAuley J, Pandey J, Leskovec J," Inferring networks of substitutable and complementary products", *Knowledge Discovery and Data Mining, 2015.*

[4] Mikolov T, Karafiat M, Burget L, Cernock L, Khudanpur S. "Recurrent neural network based language model*",* In: *Proceedings of Interspeech, 2010.*

[5] *Pascanu, R., Mikolov, T. & Bengio, Y.* "On the difficulty of training recurrent neural networks". In *Proc. 30th International Conference on Machine Learning, 2013, pp.1310–1318.*

[6] Schuster, M. and Paliwal, K. K. "Bidirectional recurrent neural networks". *Signal Processing, IEEE Transactions on, 45(11), 1997, pp. 2673–268.*

[7] Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. "Recurrent convolutional neural networks for text classification". *In Twenty-Ninth AAAI Conference on Artificial Intelligence,* 2015.

[8] Socher R, Perelygin A, J. Wu, J. Chuang, C. Manning, A. Ng, C. Potts. "Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank". *In Proceedings of EMNLP 2013.*

[9] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. "Dropout: a simple way to prevent neural networks from overfitting". *J. Machine Learning Res. 15, 2014, pp.1929–1958.*

[10] Timmaraju A, Khanna V. "Sentiment analysis on movie reviews using recursive and recurrent neural network architectures". *Semantic Scholar*; 2015.

[11] Wang, Xingyou et al. "Combination of Convolutional and Recurrent Neural Network for Sentiment Analysis of Short Texts." In *COLING 2016.*

[12] Zhou et al. "Text Classifiation improved by Integrating Bidirectional LSTM with Two-dimensional Max Pooling". *arXiv preprint* arXiv:1611.06639, 2016.

[13] Zygmunt Z. "Deep Learning Architecture Diagrams - Fastml". *Fastml.com. N.p., 2017. Web. 19 Mar. 2017.*