Applications of Neural Networks with Long Short-Term Memory

Kshitij Yelpale
High Integrity Systems
Frankfurt University of Applied Sciences,
Hessen, Germany
Matriculation number: 1322509
kshitij.yelpale@stud.fra-uas.de

Safir Mohammad Shaikh
High Integrity Systems
Frankfurt University of Applied Sciences,
Hessen, Germany
Matriculation number: 1322554
safir.shaikh@stud.fra-uas.de

***Abstract*—Recurrent Neural Networks have gained significant popularity in many fields related to sequential tasks and the key reason behind their success is LSTMs (Long Short-Term Memory). In this paper, we present our research-work regarding the same. We have studied LSTMs in depth and implemented an application called "Sentiment Analysis" to demonstrate their efficiency. We have also used a special variant of LSTM called "Bidirectional LSTM" during model development. After analyzing the data collected, we have shown that LSTMs give excellent results compared to traditional methods. Moreover, we have established that LSTM is better in analyzing the emotion of long sentences and can produce better accuracy and recall rate. It can get a complete sequence and information effectively as its computational complexity per time step is O(1). LSTMs have solved many complex and time-consuming problems that could never be solved previously by any algorithm, especially RNNs. It gains much popularity due to its memory cell state behavior which works like a human memory.**

1. Introduction

How a human reacts to the situation it depends upon the knowledge and experience. Similarly in the world of machine learning, huge data acts as an experience while training. In a traditional neural network, neurons just train the given data and output information we need. In this, it just tries to find out the pattern inside numbers. But in the field of natural language processing, it is not that easy because every word and its sequence has some meaning which is very difficult to predict. Humans have evolved linguistics over the years of practice. They can interpret the sentences by inferring from nonverbal cues and contexts. However, that's not the case with machine learning as the number of input sources and correlation among various models is still not up to the mark of an experienced human. Sometimes it is hard for humans to predict because most of the time sentences have many figures of speech, so making a machine to understand is a very tedious task. This is achieved by the family of Recurrent Neural Network. It does not train data from scratch for every input, it stores information in the memory like humans persist thoughts. It keeps track of every previous event to process the upcoming one.
Let's say you're reading reviews online to determine if you want to buy a product. Whenever you read a review online, your brain always only remembers crucial keywords. If your goal is trying to judge if a certain review is good or bad, you pick up words like 'awesome' and 'must buy item'. You don't care much for words like 'of', 'would', 'say', 'for' etc. If a friend asks you the next day what the review said you wouldn't remember it word by word. The other words would just fade away from memory apart from essential ones. And That's what an LSTM does.
We will see how RNN works and its drawbacks and how it can be solved by LSTM in detail. After that, we will deep dive into the LSTM architecture. There is one improved version of LSTM called Gated Recurrent Unit(GRU). Moreover, we will also discuss some different structures of RNN like bidirectional and deep RNN, which can also be worked with LSTM. In the end, we will see application fields and problem statements which can be mainly solved by these network architectures.

2. Recurrent Neural Networks

2.1. Hyperbolic Tangent Activation Function

$$\sigma_{tanh}(s) = \frac{sinh(s)}{cosh(s)} = \frac{e^s - e^{-s}}{e^s + e^{-s}}$$

The hyperbolic tangent tanh activation is used to help regulate the values flowing through the network. The tanh function squishes values to always be between -1 and 1. When vectors are flowing through a neural network, it undergoes many transformations due to various math operations. So imagine a value that continues to be multiplied by let's say 3. You can see how some values can explode and become astronomical, causing other values to seem insignificant. A tanh function ensures that the values stay between -1 and 1, thus regulating the output of the neural network.

2.2. Structure

There are some problems in standard neural networks in which inputs and outputs are of different lengths in different examples and they do not share weights learned

across different positions of text. All these problems can be solved in RNN. Contributed by David Rumelhart in 1986, Recurrent neural networks are a type of neural networks that have an internal memory which allows the output of previous layers to be served as input to the next layers. The simple structure of RNN contains an input state (x), hidden state and an output state (y), whereas wxh, w and why are the weights of the network. As its name indicates, we get a recurrent structure as we unfold the compressed structure.
To discuss in detail, first words get converted into vectors and then each vector is be given one by one to RNN. This vector is input to the network which applied to activation function Tanh with hidden state from previous neuron state and outputs as a hidden state for next neuron.

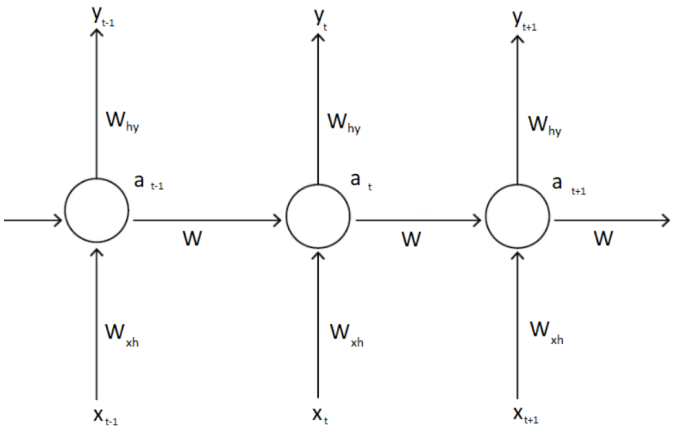

Figure 1. RNN Architecture [1]

$$h_t = tanh(W_{hh}.h_{t-1} + W_{xh}.x_t + b_h)$$

$$y_t = W_{hy}.h_t + b_y$$

where,
h_t = Hidden state,
h_{t-1} = Previous Hidden State,
tanh = Hyperbolic Tangent Activation function,
b_h, b_y = Bias,
W_{hh} = Weights at the previous hidden state,
W_{xh} = Weights at the current input state,
W_{hy} = Weights at the output state,
y_t = Output State

2.3. Usage

RNNs allow data to persist as they have loops in them and are mostly used in the field of Natural Language Processing (NLP) and Speech Recognition to recognize sequential characteristics of data and predict the future's likely scenario.
RNN is extensively used among other types in applications having sequences of data and even in convolutional layers to make the pixels effective.
In 1993, a Neural History Compressor solved a difficult deep learning task that required more than 1000 layers of RNN [2].

2.4. Problem

Despite being so powerful, RNN has several drawbacks like Gradient vanishing and exploding as it is difficult to capture long term dependencies. Therefore, they cannot apply to long sequences of data. Theoretically, RNNs perform well for long-term dependencies but practically they fail to prove it.
At each timestep T, the derivative of the loss L concerning weight matrix W is calculated:

$$\frac{\partial L^{(T)}}{\partial W} = \sum_{t=1}^{T} \frac{\partial L^{(T)}}{\partial W}$$

While performing error backpropagation, the Gradient of each node is calculated according to the gradient in the previous layer. Hence, if the previous gradient is small then the current gradient will be even smaller. This is the reason why gradients shrink as it propagates back and the initial layers cannot learn effectively due to low gradients and RNN can forget what is seen and longer sequences does having short term memory.
This problem also depends on the activation function. For sigmoid function which outputs the value in 0 and 1. Due to which when gradient shrinks contribution to initial layers becomes almost 0 and never reaches to global minima. This is called as a Gradient vanishing. In the other case of gradient exploding if the activation function ReLU, which results in output 0 for negative numbers and the same number for positive inputs, has large gradient values. So that it never reaches to global maxima because step size becomes too large for iteration.

2.5. Solution

To overcome these drawbacks of RNNs, they are modified to develop Long Short Term Memory (LSTM) networks. Created as a solution to Short-Term memory, LSTMs are needed for an hour that performs way better than other techniques for long-range dependencies. Coined by 2 computer scientists, Sepp Hochreiter and Jürgen Schmidhuber in 1997, the initial version of LSTM architecture did not include Forget gate which was added later in 1999. Many applications which could not be solved by RNNs would have remained unimplemented if LSTMs did not exist. LSTM is better in analysing emotion of long sentences and can produce better accuracy rate and recall rate. It can get a complete sequence and information effectively. They have a special mechanism called gates that regulate the flow of information. These gates learn to keep information update to date with the context of the sentence and results in an accurate prediction. All state of the art results is achieved by this improved family of RNN.

3. Long Short-Term Memory

Long Short-Term Memory Networks are abbreviated as LSTM an consist of following:

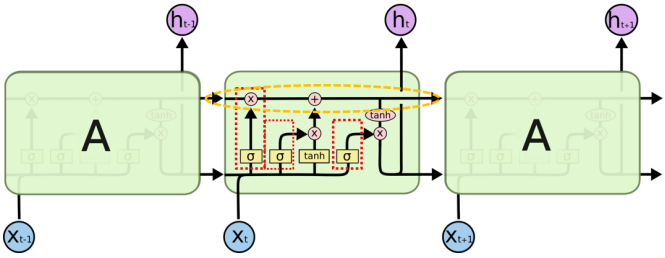

Figure 2. LSTM Architecture [3]

3.1. Structure

LSTM consists of:
The information: which is passed through the layers is in the form of vectors.
Tanh Activation function: It is used to regulate the values of a neural network.
Sigmoid activation function: It is used to update/forget the information. The more the value is closer to one, it's completely passed on and the closer to zero, it is eliminated. Sigmoid is almost similar to tanh but instead of transforming values between -1 and +1, it squishes the values between 0 and 1 so that data can be updated or forgotten since any number getting multiplied by 0 is 0 and any number multiplied by 1 is the same number so the value is kept as it is.

$$\sigma_{sigmoid}(s) = \frac{1}{(1+e^{-s})}$$

The cell state: It is called as the memory of the LSTMs. Information is added to or removed from the cell state by gates.
The gates decide what information stays or forgets by the network. Each gate in itself is a neural network.
Forget Gate: Decides what information should be kept/thrown away from previous steps.
Input Gate: Decides what new information to be added in the current step.
Output Gate: Decides next hidden state.
Having this architecture, LSTMs do not need to strive to learn the behaviour of practically remembering data for a long time as it's their default property.

3.2. Functioning

As the neural networks mimic the functioning of the brain, LSTMs mimic the functioning of human memory. Internal mechanisms of LSTMs are operated through 'Gates' that regulate the flow of data. The gates take key decisions such as which information in the sequence is crucial and which can be ignored. LSTM maintains a long chain of information in which 'the information which is stored from the beginning' is available. This information is then used for predictions.
First, data from previous hidden state and data from the current input serve as input to the sigmoid activation function. This is Forget gate. As in the ANN, we represent the output y as,

$$y = f(\sum w_i.x_i + w_0)$$

In LSTM we represent different terms similarly.

$$f_t = \sigma(W_f.[h_{t-1}, x_t] + b_f)$$

where,
f_t = Forget gate output,
σ = Sigmoid activation function,
W_f = Forget gate Weights,
h_{t-1} = Previous hidden state,
x_t = Input at time step t,
b_f = Bias

Next, an Input gate is encountered which is responsible for updating the cell state. The same input as Forget gate is passed on to the Input gate which consists of two layers. First, Sigmoid layer and second, Tanh layer. The sigmoid layer processes the input to determine which values will be updated and the tanh layer processes the input so that the flow of the network is properly regulated. Then, multiply the outputs of both layers to decide which information to keep.

$$i_t = \sigma(W_i.[h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = tanh(W_c.[h_{t-1}, x_t] + b_c)$$

where,
i_t = Input Gate output,
W_i, W_c = Input gate Weights,
b_i, bc = Bias
$\tilde{C}_t$ = Candidate

Meanwhile, the cell state is updated. The previous cell state (Ct-1) is point wise multiplied with the result of the forget gate. The result is then added with the result of Input Gate to generate a new Cell state to be passed along.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

where,
C_t = Cell State

Finally, the output gate decides the next hidden state which predicts the results based on its information from previous inputs. The same inputs as of Forget gate and Input gate is passed on to the Sigmoid function and the updated cell state is passed to Tanh function and the output of both these functions is multiplied to create a new hidden state to be passed on to the next time step

$$o_t = \sigma(W_o.[h_{t-1}, x_t] + b_o)$$

$$h_t = o_t + tanh(C_t)$$

where,
o_t = Output gate,
h_t = Current hidden state

3.3. Error Back-propagation

Now, let's see how back propagation is done in LSTMs. And you won't be surprised to know that it works similarly how it is done in RNN and ANN. As usual, We have two passes, Forward pass and Backward pass.

3.3.1. Forward Pass

In FP, we go from step 1 to T and we follow the same steps which we have seen just now, i.e.

$$f_t,\ i_t,\ \tilde{C}_t,\ o_t,\ h_t$$

3.3.2. Backward Pass

In Backward pass, for each LSTM cell we calculate the Loss function and as you can see here the partial derivative of loss function at t-1 with respect to cell state t-1 is calculated as summation of these given two terms. In first term, the loss function at t-1 is dependent upon previous hidden state and it is in turn dependent upon prev cell state. And in the second term, the loss function at time step t is dependent upon current hidden state and current hidden state is dependent upon current cell state which in turn is dependent upon previous cell state.

$$\frac{\partial L(T-1)}{\partial c_{T-1}} = \frac{\partial L(T-1)}{\partial h_{T-1}} \frac{\partial h_{T-1}}{\partial c_{T-1}} + \frac{\partial L(T)}{\partial h_T} \frac{\partial h_T}{\partial c_T} \frac{\partial c_T}{\partial c_{T-1}}$$

As in RNN, we calculate the derivative of weight vectors of all the input variables with respect to all the gates, we follow the same procedure in LSTM. As you can see we have calculated derivative of weight vectors of 2 input variables with respect to all the gates. The 1st represents for o/p gate, second for input, then forget and then finally for Cell state. Moreover, we have included the summation since we propagate the activation functions over the whole sequence. The weights are shared across the whole sequence, thus we need to take the same summation over t. [4]

$$dW_{xo} = \sum_t o_t(1-o_t)x_t do_t \quad dW_{ho} = \sum_t o_t(1-o_t)h_{t-1} do_t$$

$$dW_{xi} = \sum_t i_t(1-i_t)x_t di_t \quad dW_{hi} = \sum_t i_t(1-i_t)h_{t-1} di_t$$

$$dW_{xf} = \sum_t f_t(1-f_t)x_t df_t \quad dW_{hf} = \sum_t f_t(1-f_t)h_{t-1} df_t$$

$$dW_{xc} = \sum_t f_t(1-g_t^2)x_t dg_t \quad dW_{hc} = \sum_t f_t(1-g_t^2)h_{t-1} dg_t$$

4. Other Variants of RNN

This section comprises other variants of RNN family.

4.1. Bi-directional RNN/LSTM

The network you seen previously only works in one direction and one of the problem of that network is, we can not get the context of word in the middle sequence completely based on the words after. So that is unidirectional RNN. But in this network the blocks are in backward direction. This neural network forms a Acyclic graph. Here the information taken from past as well as from future. This network can make predictions anywhere even in the middle of a sequence with the help of information from the entire sequence.

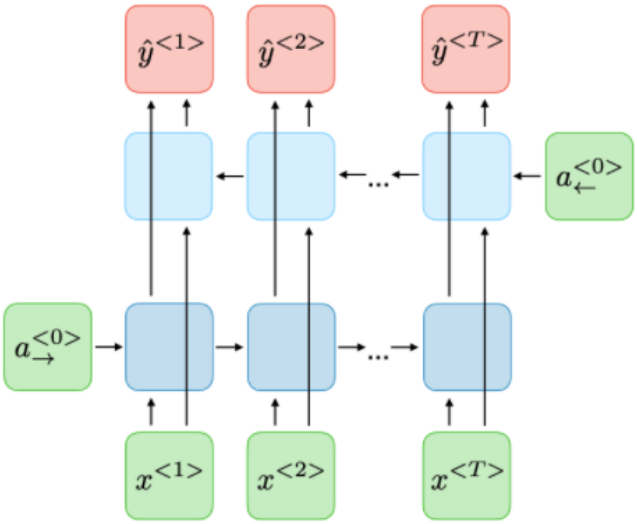

Figure 3. Bi-directional RNN Architecture [5]

The main drawback of these networks is that you need the entire sequence of data before making any predictions anywhere. For instance, while building a speech recognition system, we need to wait for the person to stop talking to get the entire utterance before you can actually process it and make a speech recognition prediction. So for a real type speech recognition applications, they're somewhat more complex. But for a lot of natural language processing applications where you can get the entire sentence all the same time, the standard bidirectional algorithm is actually very effective.

4.2. Deep RNN

For learning very complex functions sometimes it is useful to stack multiple layers of RNNs together to build even deeper versions of these models. So the input is given to first hidden layer then forwarded to next stacked hidden layer, depicted in Figure 4. Finally, one can also build deep versions of the bidirectional RNN.
Deep RNNs are quite computationally expensive to train, there's often a large temporal extent already.

4.3. Gated Recurrent Unit (GRU)

GRUs are newer generation of RNNs with quite similar structure of LSTMs. GRUs got rid of the cell state and used a hidden state to transfer information instead. It also has two gates, a reset gate and an update gate. The update gate acts similar to forget in the input gate of LSTM. It decides

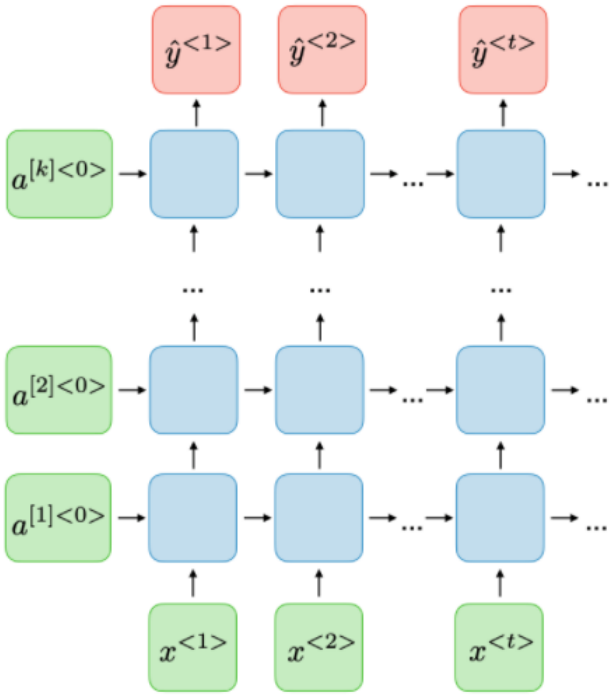

Figure 4. Deep RNN Architecture [5]

what information to neglect and what new information to add. The reset gate is a gate used to decide how much past information to forget. So that's GRU.

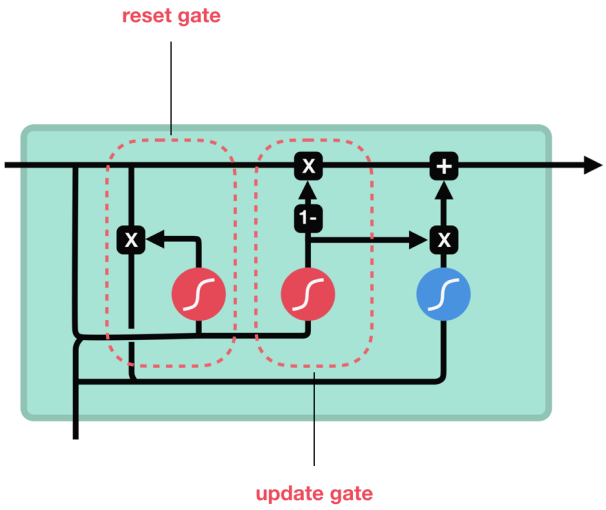

Figure 5. Gated Recurrent Unit [6]

4.4. Comparison

RNNs are good for processing sequence data for predictions but suffer from short-term memory. LSTMs and GRUs were created as a method to overcome short-term memory using a mechanism called gates. Gates act as neural networks that regulate the flow of information being passed from one-time step to the next.
GRU do use has fewer tensor operations therefore they are little speedier to train in LSTMs. Researchers and engineers usually try both to determine which one works better for their use case.

5. Application

5.1. Sentiment Analysis

With the expanding popularity of deep learning, sentiment classification is an interesting and major topic in the field of Natural Language Processing (NLP). It is a many to one architecture of RNN. A basic task in sentiment analysis is classifying the polarity of a given text whether the expressed opinion is positive, negative or neutral. More Advanced, "beyond polarity" sentiment classification looks, for instance, at emotional states such as "angry", "sad" and "happy". [7]
Initially, we developed two LSTM models which consist of

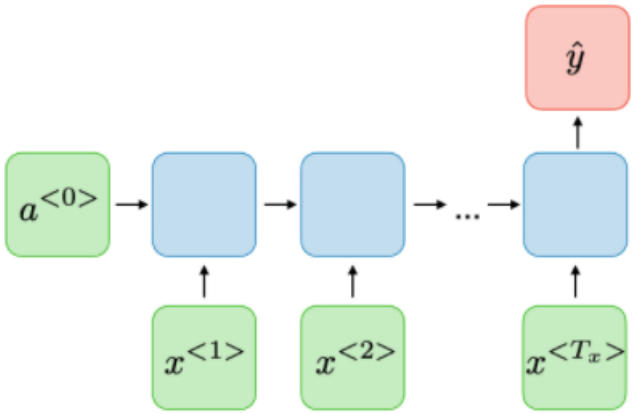

Figure 6. Sentiment Analysis: Many-to-One Network

three layers as

1) Embedding
2) LSTM
3) Dense

First, with standard and well-processed Imdb dataset provided by Keras library and the other, with custom Imdb dataset [8] having 50000 reviews. Both the dataset was well processed and well-split into training and validation data. We got 84.23% and 82.95% accuracy respectively. We used One-Hot Encoding for word embeddings in the second model. But we decided to go for a better model so that we might get better performance/results.
Then with the same datasets, we built 2 Bidirectional LSTM models from which one fulfilled our purpose but the other one did not. So, finally, we came up with a Bi-LSTM model trained on Keras' inbuilt Imdb dataset and it gave 86.76% accuracy.
Before training, we dealt with data pre-processing techniques like Removing stop words and special characters

that do not carry any meaningful information. Then, we converted the documents into lowercase and transformed the words into vectors of numbers in Imdb encoded format (Word Embedding). **This was one of our deep research on converting the words into Keras encoded Imdb vector format.** Next, we truncated or padded the documents to make a fixed length of 200 words. We have also set the vocabulary size as 20000. The model contains four layers,

1) Embedding
2) BidirectionalLSTM x 2
3) Dense

While compiling the model, we chose the best optimization algorithm called "Adam". Since Cross-Entropy is recommended for classification problems, we have used the same, i.e. "Binary Cross Entropy" in our case as we are aiming for two output classes. Also, we have saved the trained model and the corresponding weights so that we can use it later in any module by just loading the model (without the need of training again).
Moreover, we also calculated the correlation of the LSTM model predicted values with the actual sentiment values from the labelled data of 4500 documents. The Pearson correlation coefficient for LSTM = 0.882 and the Spearman correlation coefficient for LSTM = 0.886 which depicts that LSTMs are better in long-range dependencies.
Further, we also conducted a small research in which we gathered user sentiments by inviting 10 of our colleagues. From them, we collected 5 user ratings on movie reviews randomly generated from the 4500 documents of the movie review. So in total, we got 50 user sentiments and then we calculated deviations. The deviation is between LSTM predicted values and user sentiments which you can see in the following figure. Here you can see some points have deviated from the major data where mean is 0.133 and the standard deviation is 0.147.

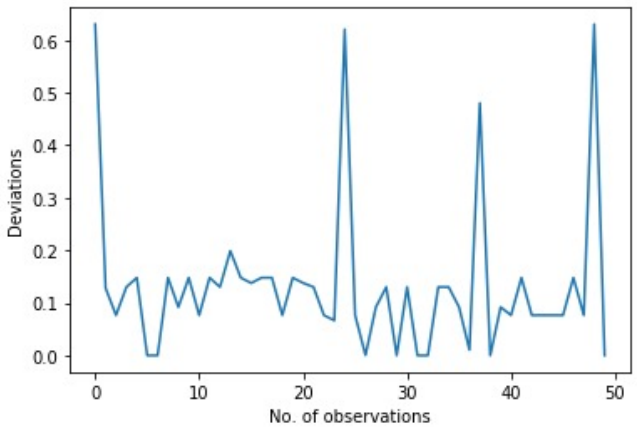

Figure 7. Plot for LSTM predicted values deviated from user ratings

The Bi-LSTM model helps to obtain the relation between the forward and backward directions of sentences. Built for Text Classification of English language long user sentiments, the model accurately extracts people's opinions from a large number of unstructured review texts and classifies them into sentiment classes.

5.1.1. Challanges

During implementation, we have faced some challanges but still we managed to successfully implement the application. Following are some of the challanges that we tried to tackle down:

- The documents having sarcasm and irony are really diffucult to classify because they are expressed alternatively.
- Definition of neutral is another challenge to tackle to perform accurate sentiment analysis
- Western and Eastern emojis play crucial role in text sentiments but difficult to identify because they are made up of special characters which sometimes are removed during data pre-processing.
- **Overfitting:** It was easily resolved by Early Stopping in which we reduced the number of epochs and stopped training as soon as it began diverging.
- **A large difference between training and validation accuracy:** We had to perform hyperparameter tuning to get rid of this issue. We added a Regularization term, thanks to the parameters 'Dropout' and 'Recurrent Dropout' provided by Keras Library.
- Between the 2 models' development, Keras updated its internal features and we had to face some challenges related to versioning and conflicts.

6. Other Examples

During recent few years, LSTMs have gained humongous popularity by its incredible and irreplaceable success in fields like Speech Recognition, Language Modeling and Translation, Image Captioning, Text Classification etc.

6.1. Speech Recognition

Unlike feed forward ANNs, RNNs have cyclic connections which make them powerful for modelling sequences. The authors have shown that LSTM based RNN architectures make more effective use of model parameters to train acoustic models for large vocabulary speech recognition [9].

6.2. Speech Synthesis

Recently, LSTM has become an attractive architecture in speech synthesis for its effect on latency and its ability to learn long time-dependencies and access past information through its recurrent connections. LSTM-RNNs produce significantly better speech than DNNs and allow a fully streaming architecture and low-latency speech synthesis [10] [11].

6.3. Text Generation

LSTMs can also be used as generative models. Not only for making predictions but they can also learn the sequences of a problem and then generate entirely new plausible sequences for the problem domain [12].

6.4. Video Captioning

By taking a video as a sequence of features, LSTM model is trained on video-sentence pairs to associate a video to a sentence. LSTM captures salient structures of video and explores the correlation between multi-modal representations (i.e., words and visual content) for generating sentences with rich semantic content [13].

6.5. Time series analysis

LSTM is a type of recurrent neural network that can learn the order dependence between items in a sequence. LSTMs have the promise of being able to learn the context required to make predictions in time series forecasting problems, rather than having this context pre-specified and fixed [14]. They model univariate time series forecasting problems consisting of a single series of observations and learn from the series of past observations to predict the next value in the sequence [15].

6.6. Name Entity Recognition

This application falls under many to many RNN architecture type where input and output sequence is the same in length. The Author proposed LSTM conditional random fields (LSTM-CRF) which is an LSTM-based RNN model that uses output-label dependencies with transition features and a CRF-like sequence-level objective function. As LSTMs are powerful for sequential data, Empirical results reveal that the models proposed by the author attain state-of-the-art performance for named entity recognition [16].

6.7. Video Activity Recognition

The authors have proposed a novel action recognition method by processing the video data using a convolutional neural network (CNN) and deep bidirectional LSTM (DB-LSTM) network. The DB-LSTM learns sequential information among frame features and processes lengthy videos by analyzing features for a certain time interval. The authors came up with significant improvements in this field as compared to other action recognition methods [17].

6.8. Image Captioning

Image captioning means generating text descriptions from images. One to many RNN architecture is used to solve this problem where there is an image as input and results into text describing an image. The authors have developed a system which extracts features from images using a CNN, combines the features with an attention model and generates captions using LSTM [18].

6.9. Machine Translation

This is many to many RNN architecture in which input and output length is different when translating sentences from one language to another. The technique of encoder-decoder has been used to have a successful translation.

6.10. Music Generation

This application comes into one to many RNN architecture, where input is one and in the output, many predicted music notes.

7. Conclusion

As the RNNs practically fail to consider long-range sequences, a special unit of RNN called LSTM achieves to do so with a lot of ease and efficiency. It is widely used in many neural network applications and it has produced extraordinary results in many fields with Time-Series data, Long Sequence data etc. In the future scope it can be used with other deep neural network too like variants of convolutional neural net. There is high scope of this network to have research with reinforcement learning and genetic algorithm as well.

8. Acknowledgement

This research was possible under the guidance and direction of Prof. Dr. Jörg Schäfer and tutor Fatima Butt. This paper is the final report for the course Learning from Data (LFD) taught in the Master's programme High Integrity Systems (HIS).

References

[1] M. Nayak, "Introduction to the architecture of recurrent neural networks (rnns)," June 2019.

[2] J. Schmidhuber, "Habilitation thesis: System modelling and optimization," 1993. [Online]. Available: ftp://ftp.idsia.ch/pub/juergen/habilitation.pdf

[3] Colah, "Understanding lstm networks," Aug. 2015. [Online]. Available: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

[4] G. Chen, "A gentle tutorial of recurrent neural network with error backpropagation," *CoRR*, vol. abs/1610.02583, 2016. [Online]. Available: http://arxiv.org/abs/1610.02583

[5] A. Amidi and S. Amidi, "Recurrent neural networks cheatsheet." [Online]. Available: https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks

[6] M. Phi, "Illustrated guide to lstm's and gru's: A step by step explanation," Sep. 2018. [Online]. Available: https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

[7] HPA, "Sentiment analysis : solutions and applications survey," Jun. 2017. [Online]. Available: https://towardsdatascience.com/sentiment-analysis-solutions-and-applications-survey-9e52d3ea2ac7#:~:text=Problemdefinition,positive,negative,orneutral

[8] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA: Association for Computational Linguistics, Jun. 2011, pp. 142–150. [Online]. Available: https://www.aclweb.org/anthology/P11-1015

[9] M. Sarma and K. K. Sarma, "Acoustic modeling of speech signal using artificial neural network," in *Advances in Systems Analysis, Software Engineering, and High Performance Computing*. IGI Global, 2015, pp. 282–299.

[10] H. S. Heiga Zen, "Unidirectional long short-term memory recurrent neural networkwith recurrent output layer for low-latency speech synthesis." [Online]. Available: https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/43266.pdf

[11] Y. K. T. Aye Mya Hlaing, Win Pa Pa, "Enhancing myanmar speech synthesiswith linguistic information and lstm-rnn," Sep. 2019. [Online]. Available: https://www.isca-speech.org/archive/SSW_2019/pdfs/SSW10_O_5-1.pdf

[12] J. Brownlee, "Text generation with lstm recurrent neural networks in python with keras," Aug. 2016. [Online]. Available: https://machinelearningmastery.com/text-generation-lstm-recurrent-neural-networks-python-keras/

[13] L. Gao, Z. Guo, H. Zhang, X. Xu, and H. T. Shen, "Video captioning with attention-based LSTM and semantic consistency," *IEEE Transactions on Multimedia*, vol. 19, no. 9, pp. 2045–2055, sep 2017.

[14] J. Brownlee, "On the suitability of long short-term memory networks for time series forecasting," May 2017. [Online]. Available: https://machinelearningmastery.com/suitability-long-short-term-memory-networks-time-series-forecasting/

[15] ——, "How to develop lstm models for time series forecasting," November 2018. [Online]. Available: https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/#:~:text=LSTMscanbeusedto,nextvalueinthesequence

[16] C. LEE, "LSTM-CRF models for named entity recognition," *IEICE Transactions on Information and Systems*, vol. E100.D, no. 4, pp. 882–887, 2017.

[17] A. Ullah, J. Ahmad, K. Muhammad, M. Sajjad, and S. W. Baik, "Action recognition in video sequences using deep bi-directional LSTM with CNN features," *IEEE Access*, vol. 6, pp. 1155–1166, 2018.

[18] D. L. j. Blaine Rister (blaine@stanford.edu), "Image captioning with attention."