FRANKFURT UNIVERSITY OF APPLIED SCIENCES

Applications of Neural Networks with Long Short-Term Memory (LSTM)

GUIDANCE-
PROF. DR. JÖRG SCHÄFER
MS. FATIMA BUTT

BY-
1. SHAIKH SAFIR MOHAMMAD (1322554)
2. YELPALE KSHITIJ (1322509)

Introduction

- Problems with traditional neural network:
  - Inputs and outputs are of different lengths in different examples, e.g. Machine Translation
  - They do not share features learned across different positions of text.
- What is Recurrent Neural Network ?
  - Neurons in layer *i* get feed forward input and receive input from neurons in higher (following) layers, usually next layer *i*+1 (backward), but also from neurons of same layer *i*.
  - Widely used in natural language processing.

Structure of RNN

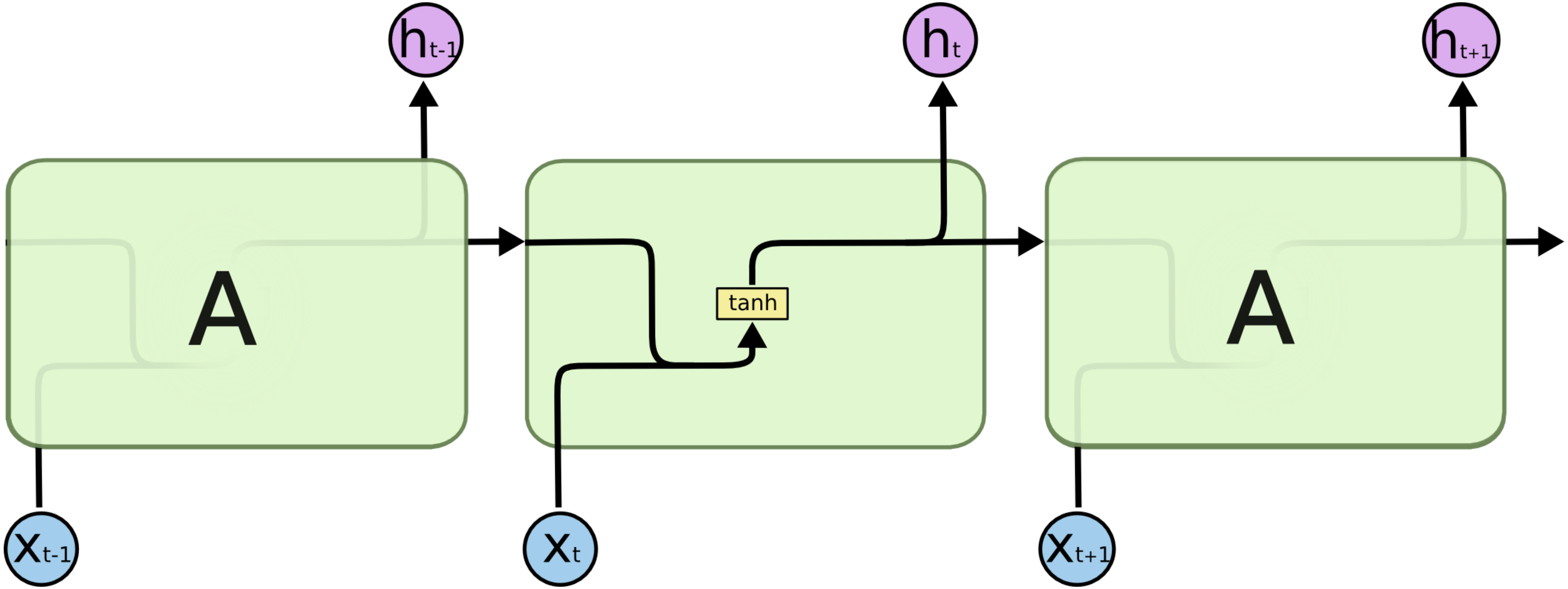
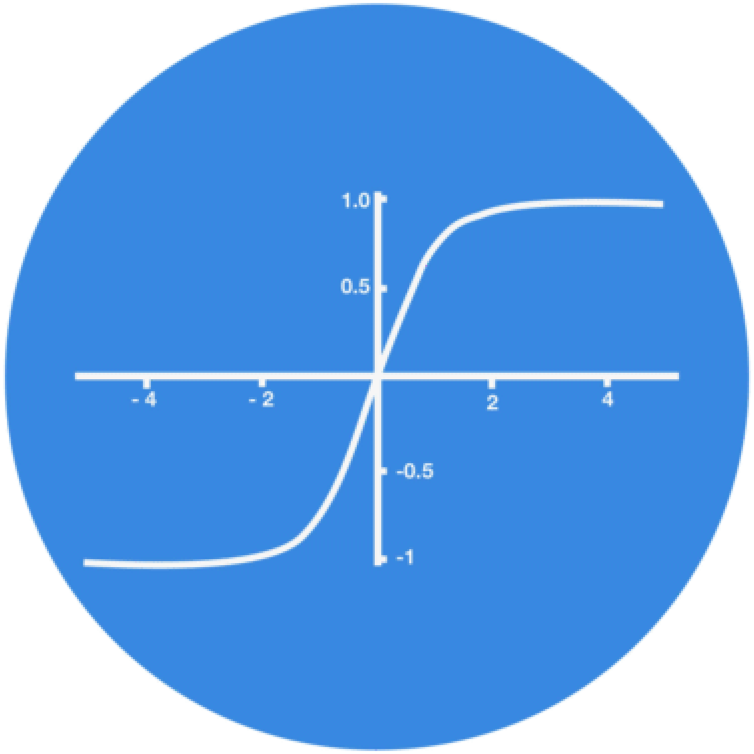


Fig: RNN Architecture [1]

Activation function – Hyperbolic Tangent

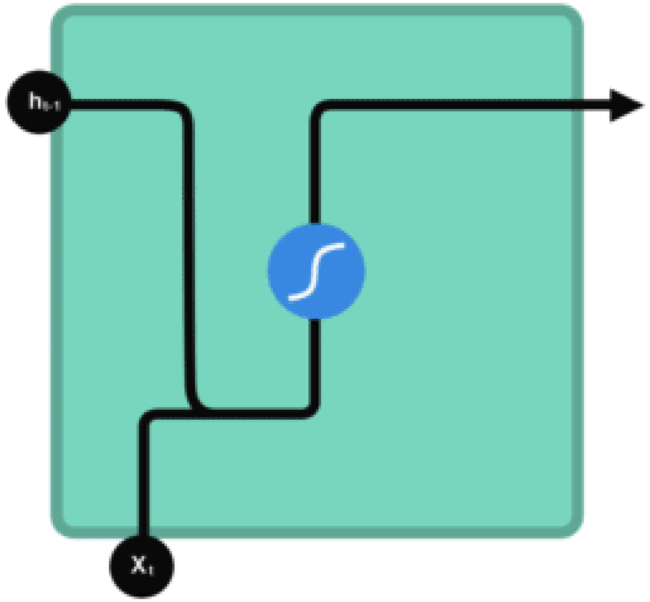
5
0.1
-0.5



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

Fig: Tanh Activation Function [2]

Structure of RNN Cell



Tanh function

new hidden state

previous hidden state

input

concatenation

$$\mathbf{h}_t = tanh(W_{hh}\mathbf{h}_{t-1} + W_{xh}\mathbf{x}_t + \mathbf{b}_h)$$ [3]

Fig: RNN Cell [2]

Gradient Problem in RNNs

- What is Gradient?
- BPTT
- 2 major gradient problems in standard RNN structure:
  - Vanishing gradient
  - Exploding gradient
- Learning algorithm can not reach at global minima in both problems

Vanishing Gradient

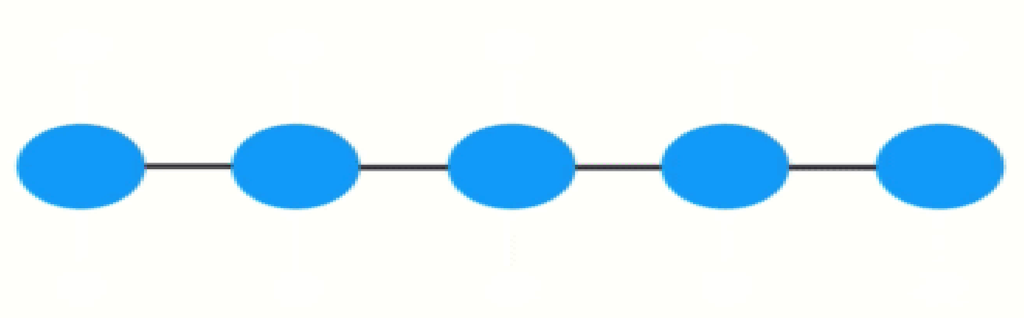


Fig: Vanishing Gradient [2]

- Gradient get minimal during back-propagation causing no effect on the learning of initial layers.
- New weight = old weight – learning rate * gradient
- 10.999 = 10.1 – 0.001
- High chances In case of Sigmoid activation function. So Tanh is preferred one!

Exploding Gradient

- Exploding gradient tends to be the bigger problem with training RNNs, although when exploding gradients happens, it can be **catastrophic** because the exponentially large gradients can cause our parameters to become so large that your neural network parameters get really messed up
- In case of ReLU function, gradient can be > 1 and on every successive layer it increases exponentially
- But exploding gradients are easier to spot because the parameters just blow up and you might often see NaNs, or not a numbers, meaning results of a numerical overflow in your neural network computation.

Solutions to Gradient Problem

- Exploding gradient
  - Truncated Backpropagation
  - Penalties
  - Gradient Clipping
- Vanishing gradient
  - Weight Initialization
  - Echo State Networks
  - Long Short-Term Memory Networks (LSTM)

LSTMs mimic Human Memory

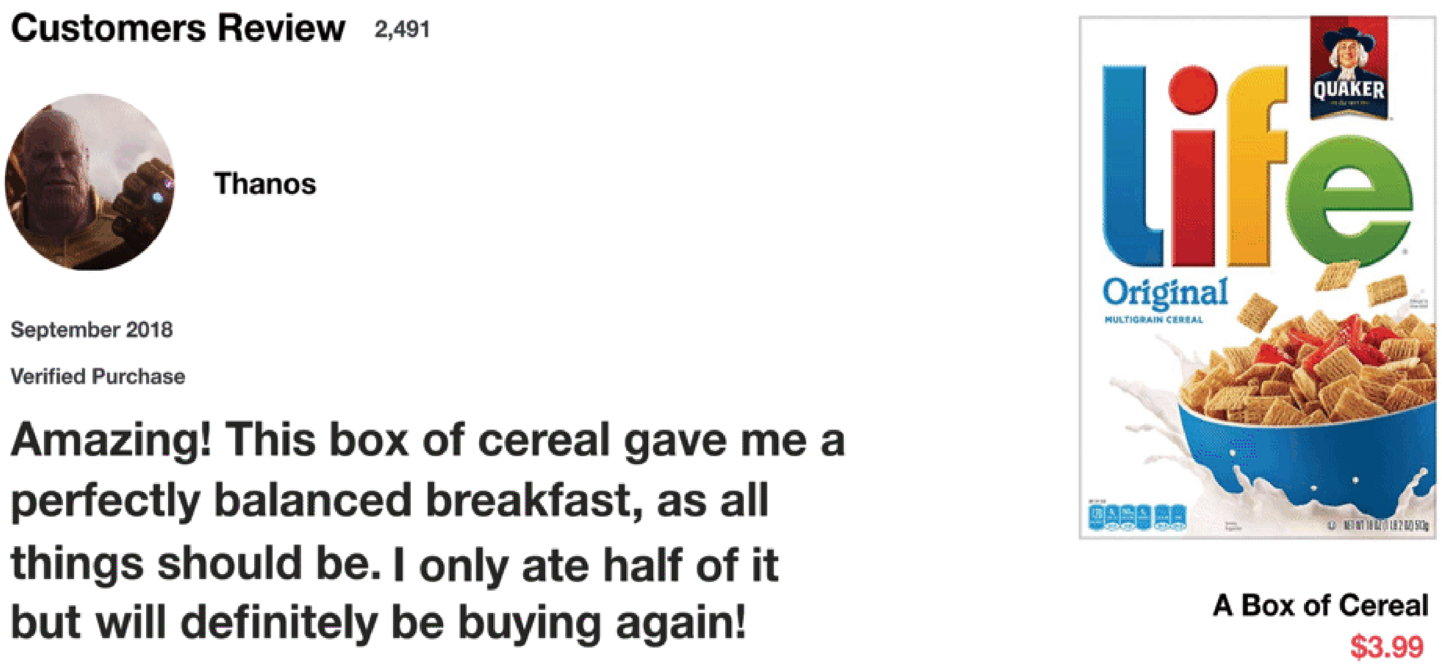


Fig: LSTMs mimic Human Memory [2]

LSTM Architecture

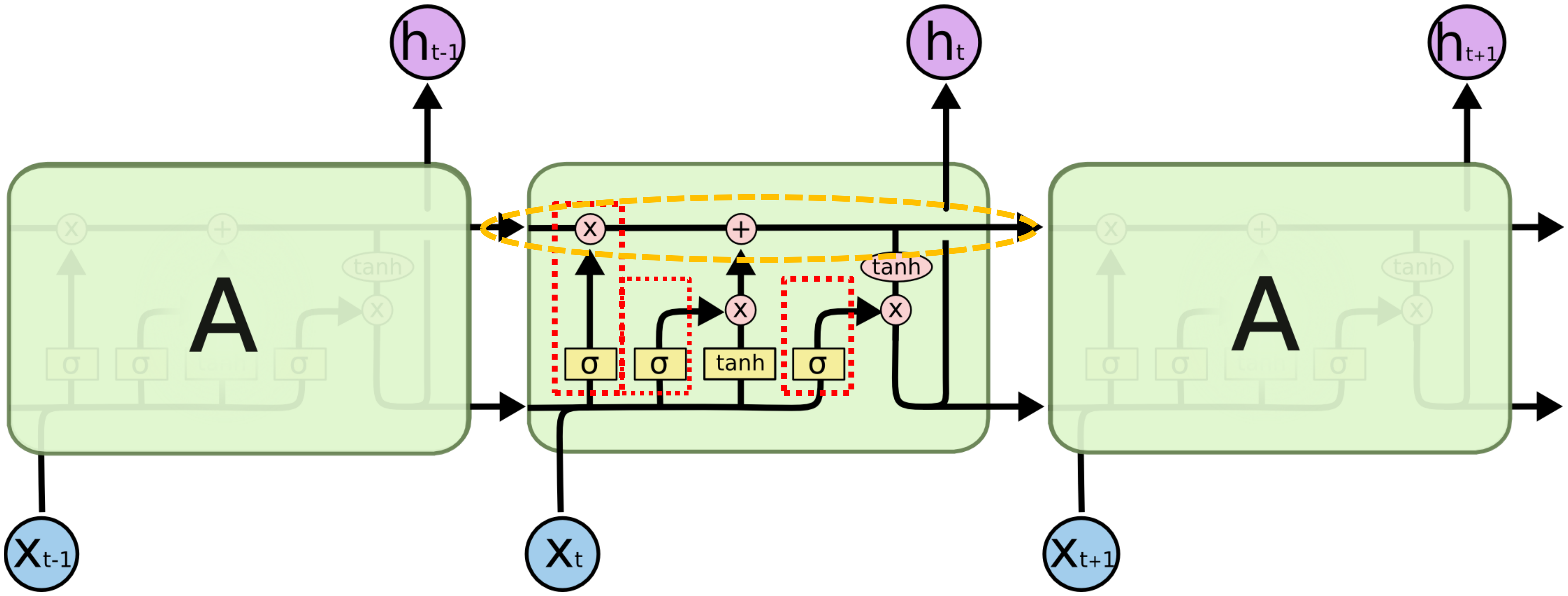
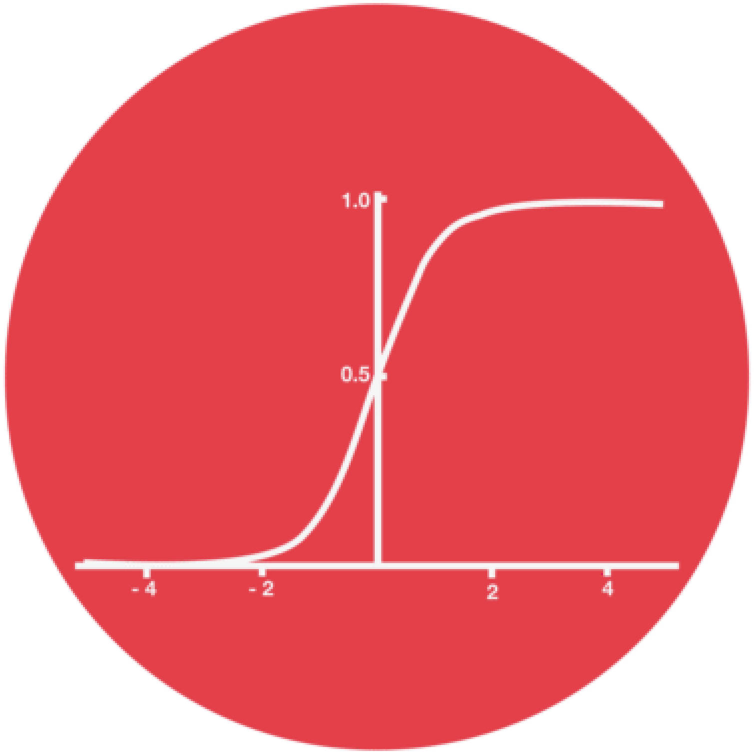


Fig: LSTM Architecture [1]

Activation function – Sigmoid

5
0.1
-0.5



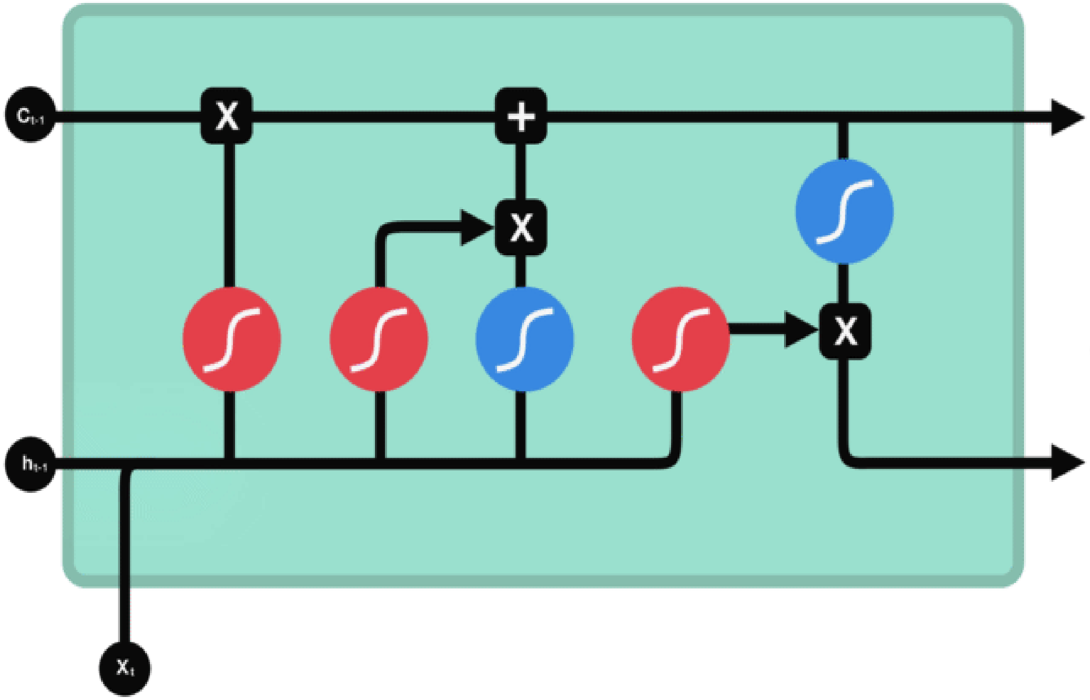
$$\sigma_{sigmoid}(s) = \frac{1}{(1 + e^{-s})}$$

Fig: Sigmoid Activation Function [2]

Forget Gate

C_{t-1} previous cell state

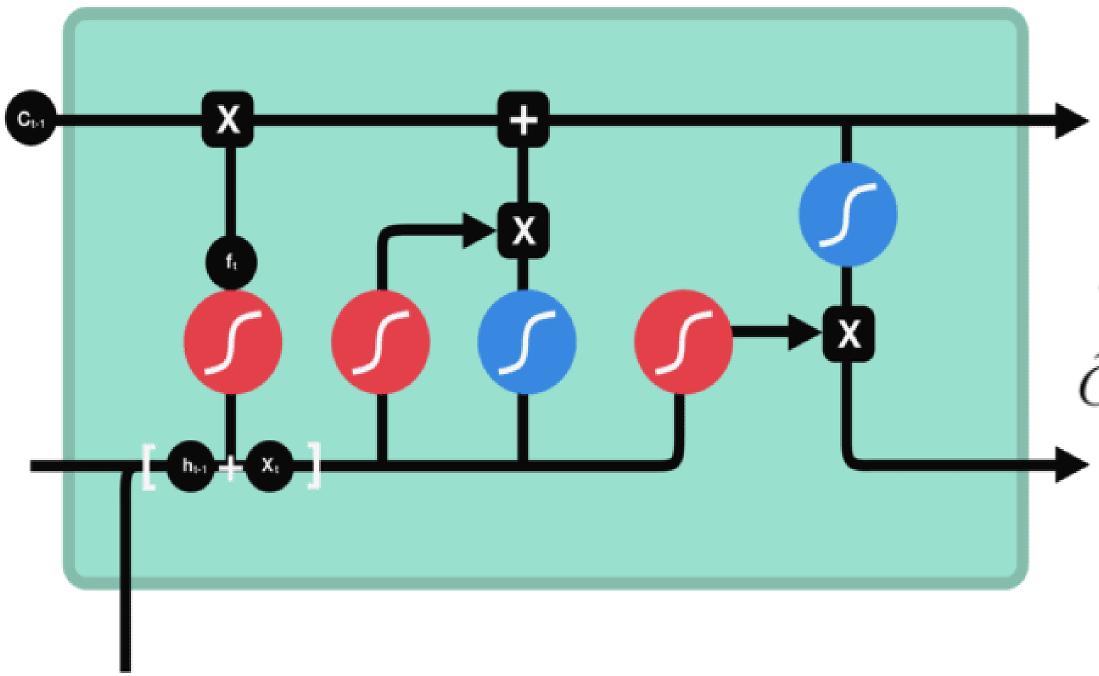
f_t forget gate output



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

Fig: Forget Gate [2]

Input Gate



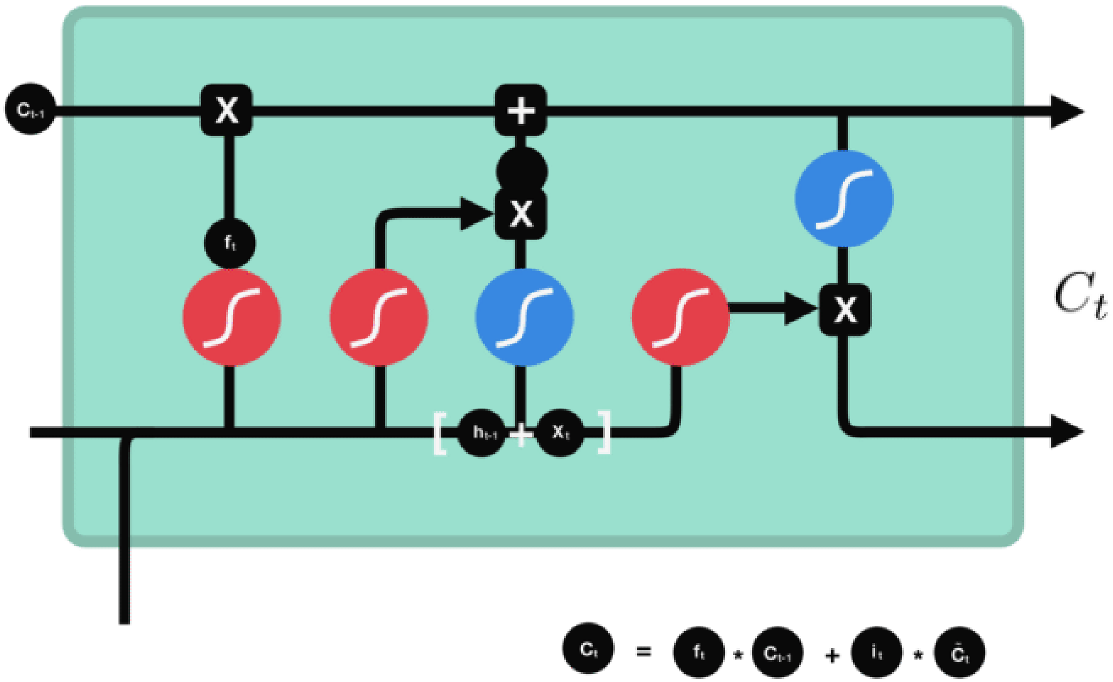
previous cell state
forget gate output
input gate output
candidate

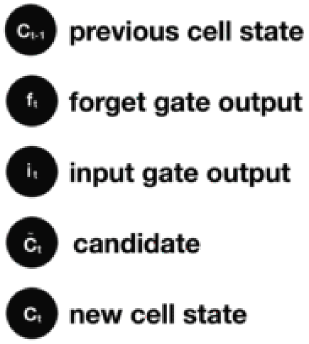
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Fig: Input Gate [2]

Cell State

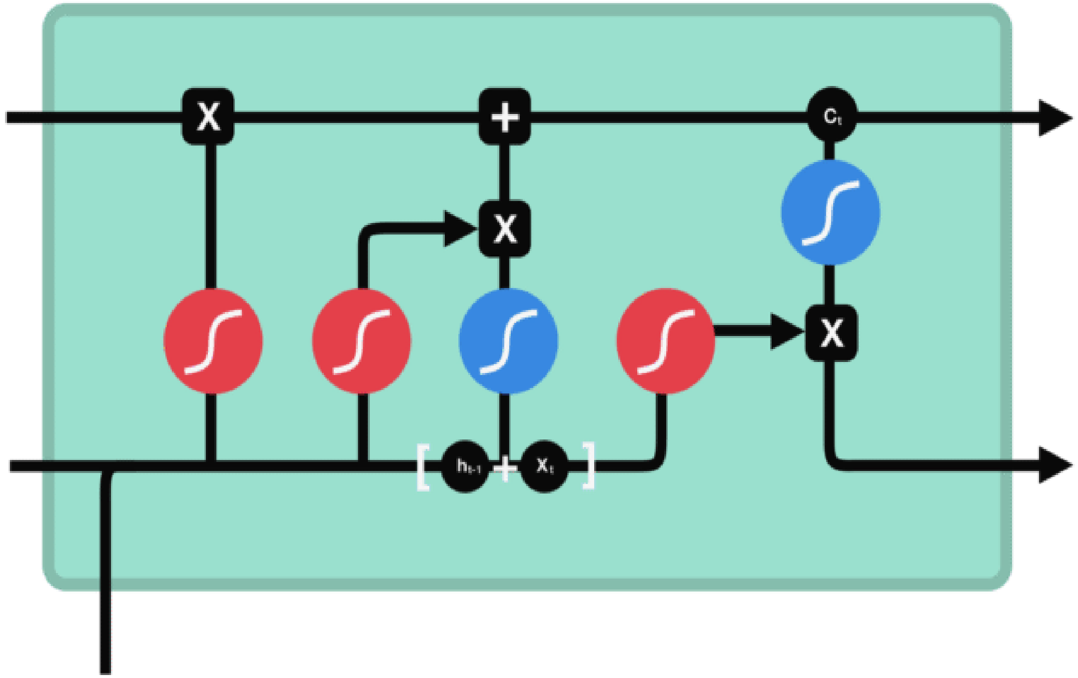




$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Fig: Cell State [2]

Output Gate



- C_{t-1} previous cell state
- f_t forget gate output
- i_t input gate output
- $\tilde{C}_t$ candidate
- C_t new cell state
- o_t output gate output
- h_t hidden state

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

$$h_t = o_t * \tanh\left(C_t\right)$$

Fig: Output Gate [2]

Backpropagation in LSTMs

- Similar to RNN and ANN back-propagation algorithms.
- 2 Passes:
  - Forward Pass
    - Step 1 to T ➜ $\boxed{f_t \;\; i_t \;\; \tilde{C}_t \; C_t \;\; o_t \;\; h_t}$
  - Backward Pass
    - Step T to 1 ➜ $$\boxed{\frac{\partial \mathcal{L}(T-1)}{\partial \mathbf{c}_{T-1}} = \frac{\partial \mathcal{L}(T-1)}{\partial \mathbf{h}_{T-1}} \frac{\partial \mathbf{h}_{T-1}}{\partial \mathbf{c}_{T-1}} + \frac{\partial \mathcal{L}(T)}{\partial \mathbf{h}_T} \frac{\partial \mathbf{h}_T}{\partial \mathbf{c}_T} \frac{\partial \mathbf{c}_T}{\partial \mathbf{c}_{T-1}}}$$ [3]

Backpropagation in LSTMs

- Back-propagate the activation functions over the whole sequence
- The weights W_{xo}, W_{xi}, W_{xf} and W_{xc} are shared across the whole sequence, thus we need to take the same summation over t

[3]

$$dW_{xo} = \sum_t \mathbf{o}_t(1 - \mathbf{o}_t)\mathbf{x}_t d\mathbf{o}_t$$

$$dW_{xi} = \sum_t \mathbf{i}_t(1 - \mathbf{i}_t)\mathbf{x}_t d\mathbf{i}_t$$

$$dW_{xf} = \sum_t \mathbf{f}_t(1 - \mathbf{f}_t)\mathbf{x}_t d\mathbf{f}_t$$

$$dW_{xc} = \sum_t (1 - \mathbf{g}_t^2)\mathbf{x}_t d\mathbf{g}_t$$

$$dW_{ho} = \sum_t \mathbf{o}_t(1 - \mathbf{o}_t)\mathbf{h}_{t-1} d\mathbf{o}_t$$

$$dW_{hi} = \sum_t \mathbf{i}_t(1 - \mathbf{i}_t)\mathbf{h}_{t-1} d\mathbf{i}_t$$

$$dW_{hf} = \sum_t \mathbf{f}_t(1 - \mathbf{f}_t)\mathbf{h}_{t-1} d\mathbf{f}_t$$

$$dW_{hc} = \sum_t (1 - \mathbf{g}_t^2)\mathbf{h}_{t-1} d\mathbf{g}_t$$

Pseudo Code

```
def LSTMCELL(prev_ct, prev_ht, input):
    combine = prev_ht + input
    ft = forget_layer(combine)
    candidate = candidate_layer(combine)
    it = input_layer(combine)
    Ct = prev_ct * ft + candidate * it
    ot = output_layer(combine)
    ht = ot * tanh(Ct)
    return ht, Ct



ct = [0, 0, 0]
ht = [0, 0, 0]

for input in inputs:
    ct, ht = LSTMCELL(ct, ht, input)
```

Other variants – Bi directional RNN/LSTM

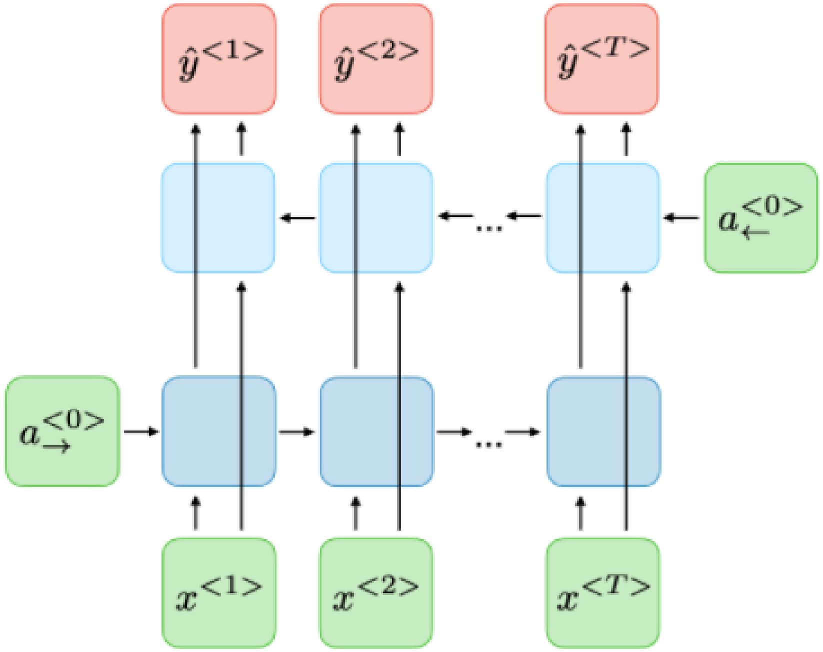


Fig: Bi – Directional Variant [4]

Other variants – Deep RNN/LSTM

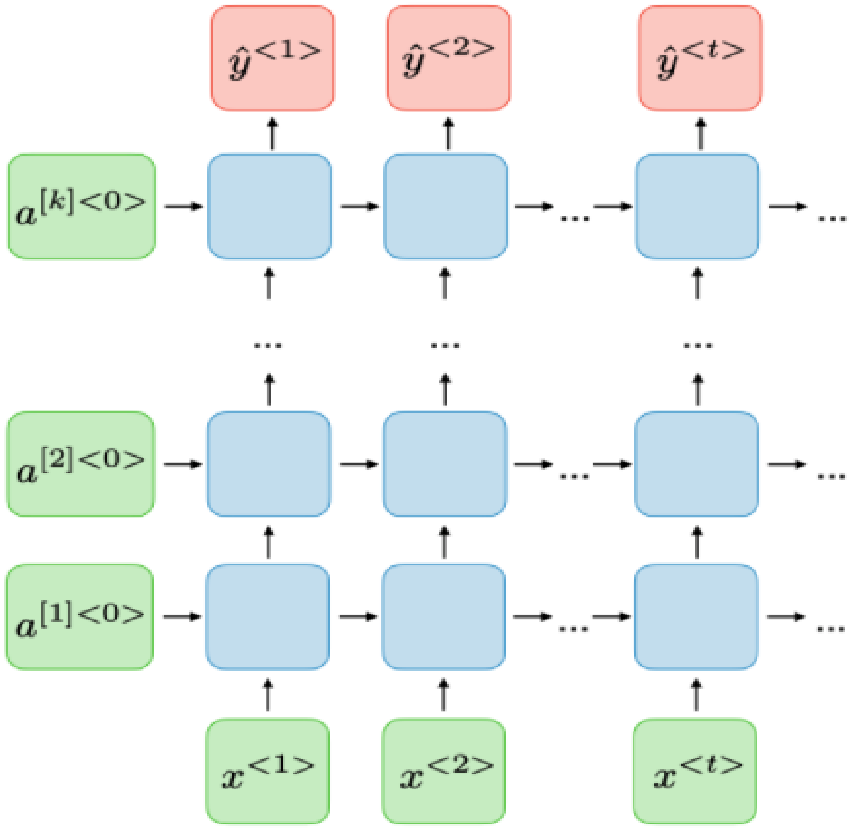


Fig: Deep RNN/LSTM [4]

Other variants – Gated Recurrent Unit (GRU)

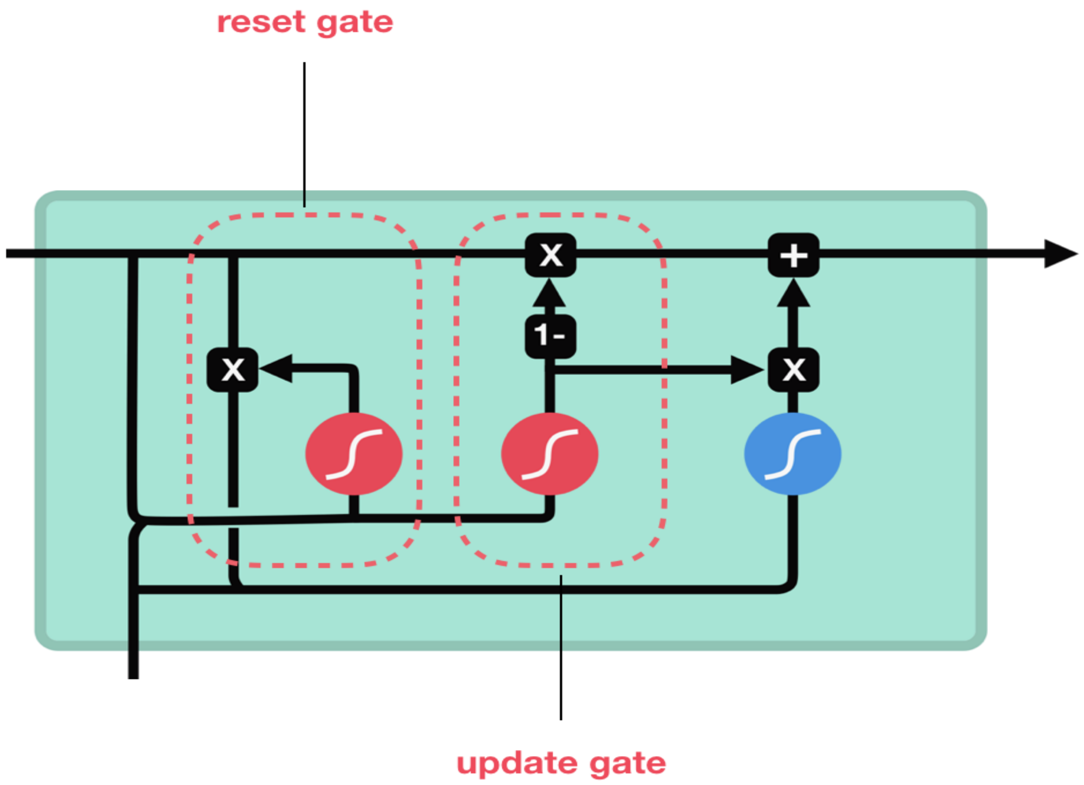
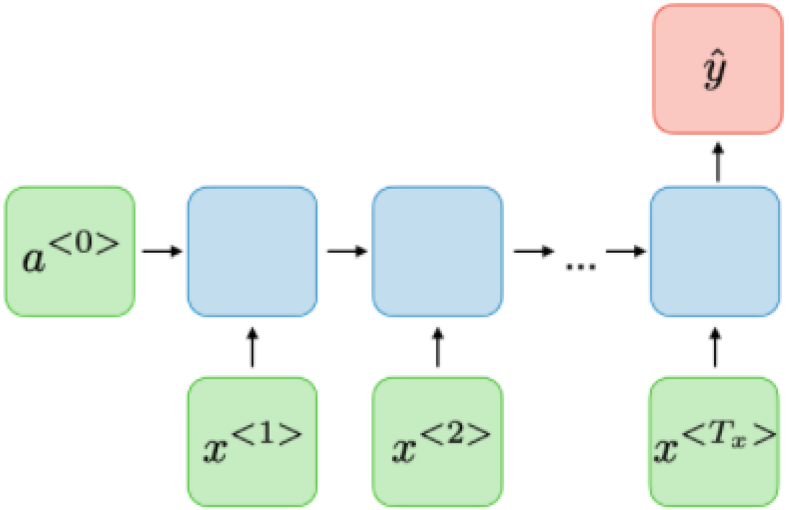


Fig: GRU Cell [2]

Application – Sentiment Analysis

- LSTM is better in analyzing emotion of long sentences and can produce better accuracy and recall rate [5]
- Sentiment Analysis is -
  - the term which is used very often in web-based business sites, social networks and different fields.
  - an examination field to analyze people's subjective sentiments.
- More Advanced, Focus on "Beyond Polarity"



[4]

Implementation

- Dataset: IMDB Movie Reviews [6]
- Data Pre-processing
- Library: Keras
- Word Embedding: One-Hot representation + Keras encoded Imdb format
- Model: (Embedding + Bidirectional(LSTM( )) + Bidirectional(LSTM( )) + Dense)

FRANKFURT UNIVERSITY OF APPLIED SCIENCES

Performance Measures

	# Epochs	Training Accuracy	Validation Accuracy
LSTM (Trained on Custom IMDB Dataset with One-Hot Encoding)	15	99.85 %	82.96 %
LSTM (Trained on Keras' IMDB Dataset)	3	93.12 %	85.96 %
Bi-LSTM (Trained on Keras' IMDB Dataset)	2	89.98 %	**86.89 %**

Demo

Challenges

- Context
  - Irony
  - Sarcasm
  - Emojis (:D, ¯ \ _ (ツ) _ / ¯ etc.)
  - Definition of Neutral
- Overfitting
- Large difference between training and validation accuracy
- Keras' feature updates!

Results - Correlation

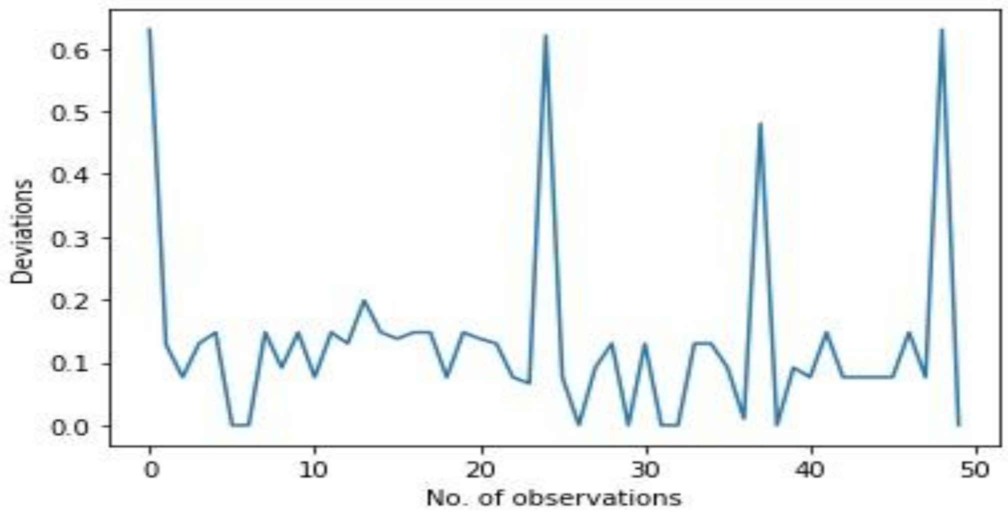
Performed correlation test –

- between predicted values of LSTM and actual sentiments
- on 4500 documents (sentiments)

Correlation Coefficient Test	LSTM and Actual sentiment
Pearson	0.882
Spearman Rank	0.886

Results – Characteristic Values

Deviations	LSTM and User sentiment
Mean	0.133
Standard Deviation	0.147



Other Applications

1. Speech recognition
2. Speech synthesis
3. Text generation
4. Video captioning
5. Time series analysis
6. Name entity recognition
7. Video activity recognition
8. Image captioning
9. Machine translation
10. Music generation

References

[1] Colah, 2015. Understanding LSTM Networks. [Blog] Colah's Blog, Available at: <http://colah.github.io/posts/2015-08-Understanding-LSTMs/> [Accessed 11 September 2020].

[2] Phi, M., Medium. 2018. Illustrated Guide to LSTM’s and GRU’s: A step by step explanation. [Blog] Available at: <https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21> [Accessed 11 September 2020].

[3] [cs.LG] Chen *, G., 2018. A Gentle Tutorial Of Recurrent Neural Network With Error Backpropagation. Department of Computer Science and Engineering, SUNY at Buffalo.

[4] Amidi, A. and Amidi, S., n.d. CS 230 - Recurrent Neural Networks Cheatsheet. [online] Stanford.edu. Available at: <https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks> [Accessed 14 September 2020].

[5] Hochreiter, Sepp & Schmidhuber, Jürgen. (1997). Long Short-term Memory. Neural computation. 9. 1735-80. 10.1162/neco.1997.9.8.1735.

[6] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. (2011). Learning Word Vectors for Sentiment Analysis. The 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011).

FRANKFURT UNIVERSITY OF APPLIED SCIENCES

Thank you!