Recurrent Neural Networks

Mengjun Wang, Tianhao Wu

THE UNIVERSITY OF TENNESSEE KNOXVILLE

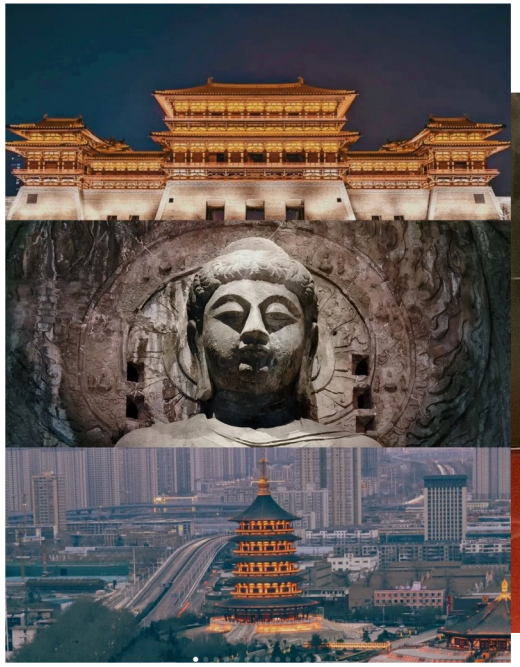
Test Questions

- Type of RNNs and list one of their applications for each.
- The main issue of vanilla RNN and raise one solution.
- List main mechanisms in LSTM (four).

TIANHAO WU

MoSIS LAB
MOBILE SENSING AND INTELLIGENCE SECURITY

- First year Ph.D. student majoring in computer science
  - Advisor: Dr. Jian Liu
  - I work on some fancy projects on fitness using mobile sensing like treadmill and cycling.
- Hometown: Henan, China



Life & Hobby

Mengjun Wang

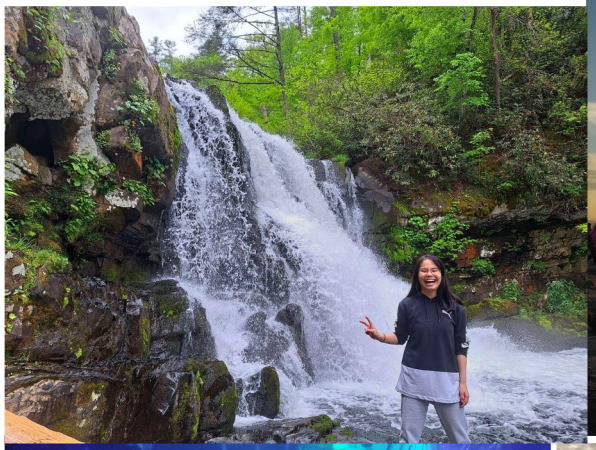
- Second year Ph.D. student in Civil Engineering.

- B.S. in Traffic Engineering. 2016-2020
- Worked as primary school teacher in China. 2020-2021
- Joined NIUX Lab in 2021 Fall, advised by Dr. Shuai Li.
- Research: AI-aided civil engineering like infrastructure detection using DL, industrial robots' application in construction, etc. (not yet decided on the dissertation direction yet.)
- Could find me on LinkedIn: https://www.linkedin.com/in/mengjun-wang-50b58525a/

Changsha: earned my bachelor's degree here

Xiangtan: my hometown

About Me



Dog Lover!

ISTJ-A

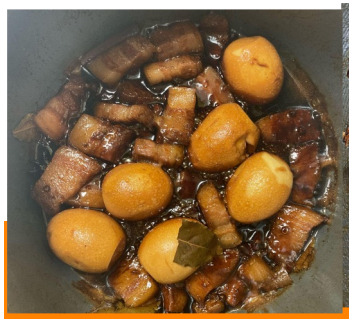






Food Lover! 🌶️








Mengjun Wang

- Anime Lover —— Detective Conan, One Piece, Crayon Shin-chan…

- KPOP Lover —— Mino, Zico, PO!!!
- Love Watching Korean Variety Show
  —— New Journey to the West, Great Escape…
- Love Playing Mahjong and Badminton.

Overview:

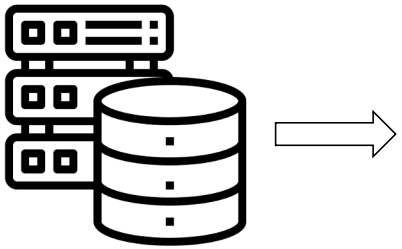
- ❑ History
- ❑ Algorithms
  - ❑ Traditional RNN
  - ❑ LSTM
- ❑ Applications
- ❑ Implementations
- ❑ Open Issues
- ❑ References

History

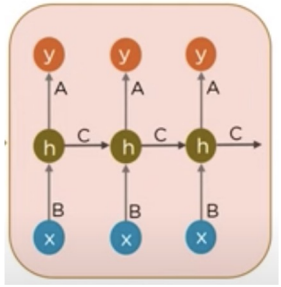
- **1982 - Hopfield Networks**: John Hopfield introduced Hopfield networks, a type of recurrent neural network where all connections are symmetric. Hopfield networks serve as content-addressable ("associative") memory systems with binary threshold units.
- **1986 - Backpropagation Through Time (BPTT)**: Paul Werbos proposed the BPTT algorithm, which is essentially an application of the chain rule of calculus to compute the gradient of the loss function with respect to the weights of the network. It's a key algorithm for training RNNs.
- **1997 - Long Short-Term Memory (LSTM):** Sepp Hochreiter and Jürgen Schmidhuber proposed LSTMs to overcome the vanishing gradient problem that plagued the training of traditional RNNs.
- **2014 - Gated Recurrent Units (GRUs)**: Kyunghyun Cho et al. introduced GRUs, a variant of LSTMs with a simplified gating mechanism and fewer parameters. GRUs have been shown to perform comparably to LSTMs on certain tasks.

Algorithm

Do you know how the Google autocomplete your search query?

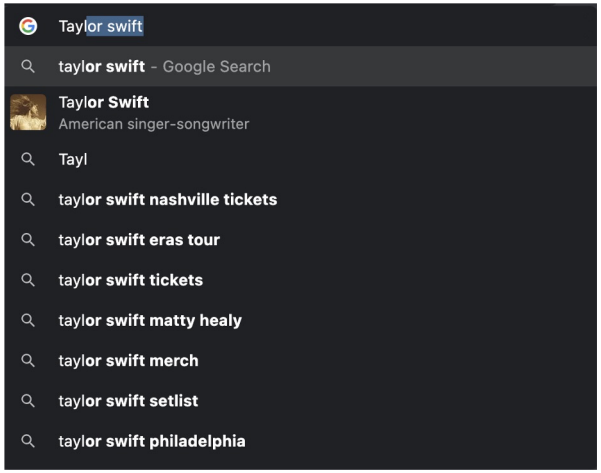


Collection of large number of most frequently occurring consecutive words.



Fed into a Recurrent Neural Network

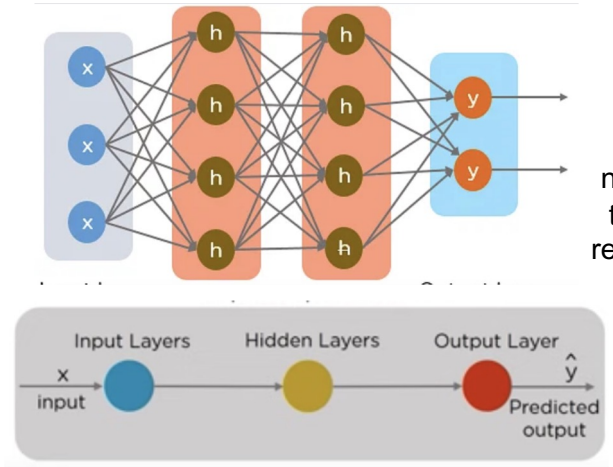
Analyses the data by finding the sequence of words occurring frequently and builds a model to predict the next word in the sequence



Autocomplete the sentence

Algorithm

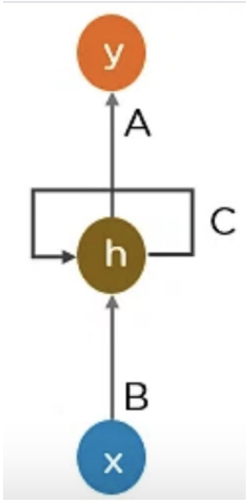
Issues in Feed Forward Neural Network:



Feed Forward Neural Network

Decisions are based on current input.
No memory about the past.
No future scope.

The nodes in different layers of the neural network are compressed to form a single layer of recurrent neural networks.



Recurrent Neural Network

A, B, and C are the parameters of the network.

A recurrent neural network (RNN) is a type of artificial neural network which uses sequential data or time series data.

RNN works on the principle of saving the output of a particular layer and feeding this back to the input in order to predict the output of the layer.

Characteristic:

Can handle sequential data.
Consider the current input and also the previously received inputs.
Can memorize previous inputs due to its internal memory.

Image from https://www.simplilearn.com/tutorials/deep-learning-tutorial/rnn

Algorithm

For a simple RNN:

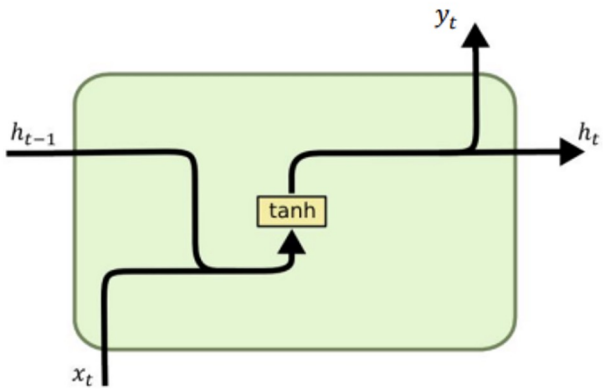
$$h_t = f_w(h_{t-1}, x_t)$$

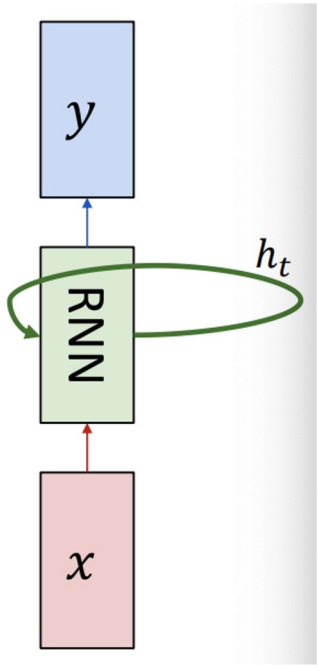
$$\downarrow$$

$$h_t = \tanh(w_{hh} \cdot h_{t-1} + w_{xh} \cdot x_t)$$

$$y_t = w_{hy} \cdot h_t$$

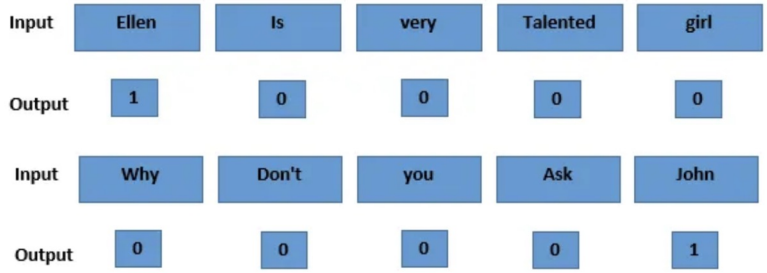
h_t: new state.
f_w: function with parameter w.
h_t-1: old state.
x_t: input vector at time step t.



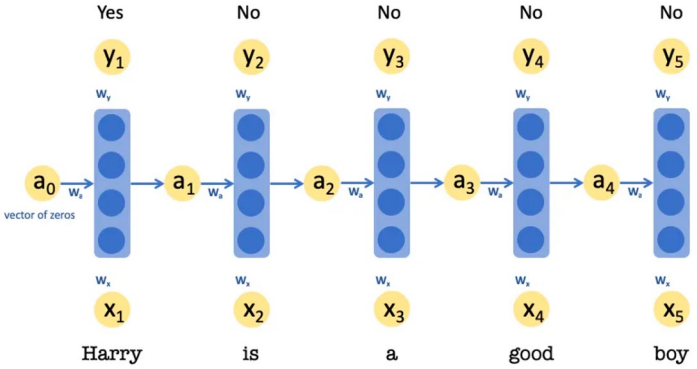


How RNN works?

A NLP example: detect names in a sentence:



If the word is name(John, Ellen …) we map it to 1.



RNN architecture

Forward propagation

5 words —-> 5 steps

for each step t we calculate a, y using the shared weights Wa,Wx, Wy, ba, by :

$a_1 = \tanh(w_a*a_0+w_x*x_1+b_a)$

$\hat{y}_1 = \text{sigmoid}(w_y*a_1+b_y)$

$a_2 = \tanh(w_a*a_1+w_x*x_2+b_a)$

$\hat{y}_2 = \text{sigmoid}\ (w_y*a_2+b_y)$

.

.

.

$a_5 = \tanh(w_a*a_4+w_x*x_5+b_a)$

$\hat{y}_5 = \text{sigmoid}\ (w_y*a_5+b_y)$

$a_t = \tanh(w_a*a_{t-1}+w_x*x_t+b_a)$

$\hat{y}_t = \text{sigmoid}(w_y*a_t+b_y)$

Cost function to represent the relation between the real output y and the output predicted ŷ for each time step t:

$\text{Cost}(\hat{y}_t,y_t) = -y_t*\log(\hat{y}_t) - (1-y_t)\log(1-\hat{y}_t)$

sum over the cost function of each word to calculate the loss function

$\text{Loss}(\hat{y},y) = \sum_{t=1}^{5} \text{Cost}(\hat{y}t, yt)$

Back propagation

Compute derivative of the loss function with respect to parameters Wa, Wx, Wy, ba, by using the chain rule to simplify the calculus.

$w_a = w_a - \frac{\alpha*\partial Loss(\hat{y},y)}{\partial Wa}$ $w_x = w_x - \frac{\alpha*\partial Loss(\hat{y},y)}{\partial Wx}$ $w_y = w_y - \frac{\alpha*\partial Loss(\hat{y},y)}{\partial Wy}$ $b_a = b_a - \frac{\alpha*\partial Loss(\hat{y},y)}{\partial ba}$

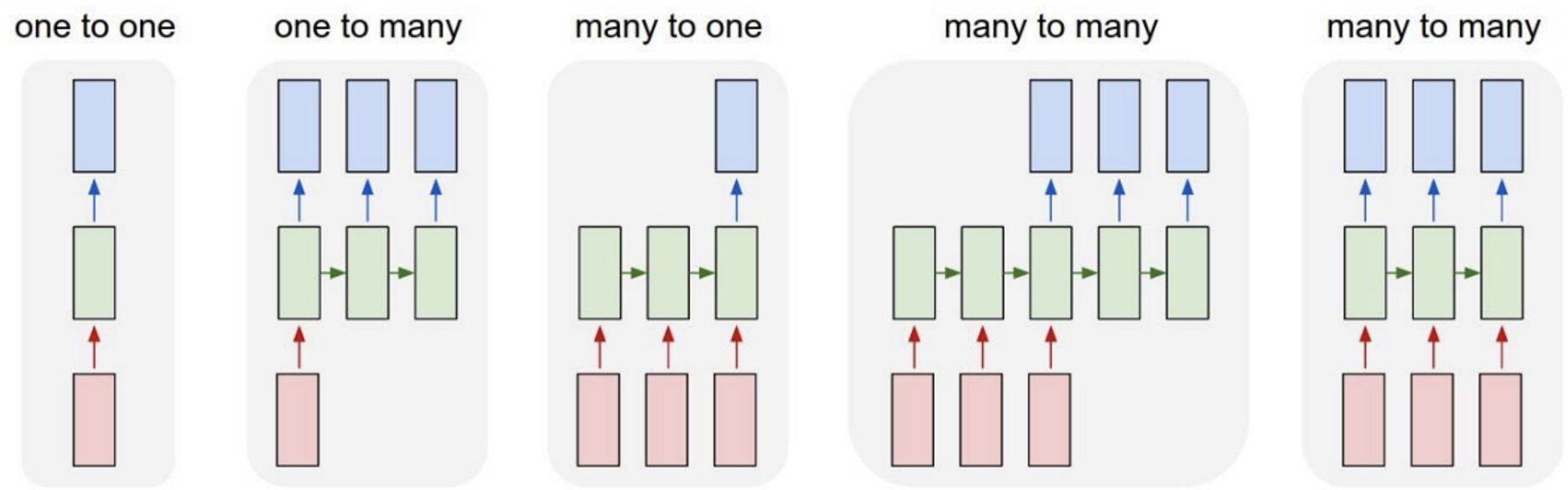
$b_y = b_y - \frac{\alpha*\partial Loss(\hat{y},y)}{\partial by}$

Minimize the loss function —> the predicted output would converge to the real output. Use the optimized weights to detect names through future sentences.

Adapted from https://medium.com/swlh/simple-explanation-of-recurrent-neural-network-rnn-1285749cc363

Type of RNN

RNNs allow us to operate on sequence inputs (input sequences, output sequences or both)



Vanilla neural network. Regular machine learning problems

Sequence of data outputs. Image captioning

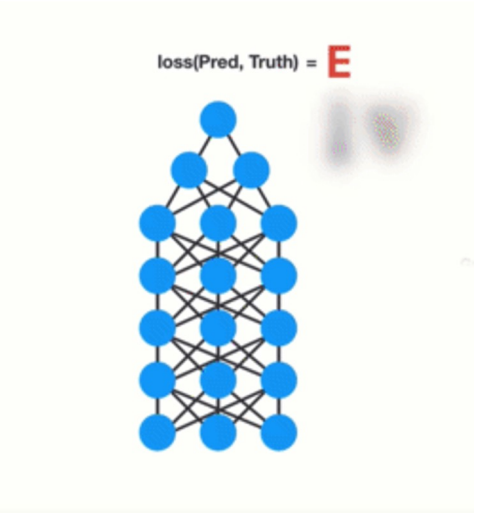
Sequence inputs. Sentence emotion classification

Sequence inputs and outputs. Machine translation.

Synced sequence input and output. Video Classification: label each frame for the video.

Image from http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Issue for Vanilla RNN



Vanishing gradient problem:

Gradients exponentially shrink as it back propagates down.
The earlier layers fail to do any learning as the internal weights are barely being adjusted due to extremely small gradients.

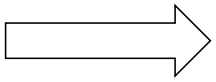
In RNN:

Think of each time step in a recurrent neural network as a layer.

The gradient values will exponentially shrink or explode as it propagates through each time step.

Because of vanishing gradients, the RNN doesn't learn the long-range dependencies across time steps.

So not being able to learn on earlier time steps causes the network to have a short-term memory.



LSTM's: Long Short Term Memory
GRU's: Gated Recurrent Unit

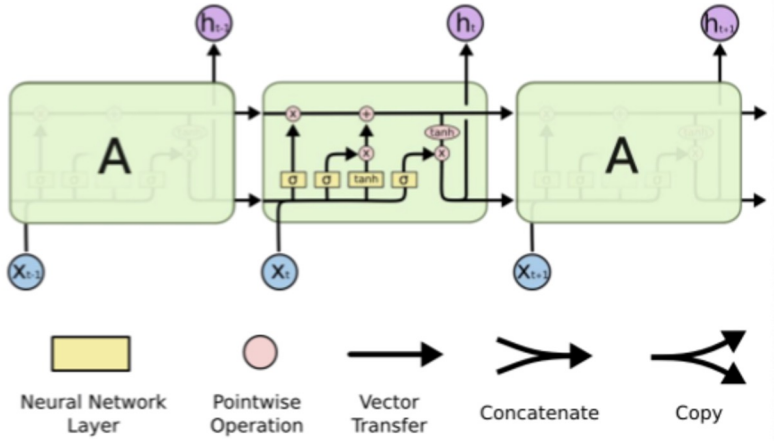
LSTM

1. First proposed in 1997 byHochreiter & Schmidhuber (1997) : http://www.bioinf.jku.at/publications/older/2604.pdf
2. Tries to deal with the problem that simple RNN's cannot learn long term dependencies.
3. Seem to work very well on a large variety of problems.
4. By default is able to remember information for long periods of time.

Name comes from the idea that we would like to extend the short term memory (memory of recent events). The leaned weights are usually thought as long term memory.

Mechanism:

- Forget Gate
- Input Gate
- Output Gate
- Cell State



LSTM-Forget Gate

Decide How Much Past Data It Should Remember.

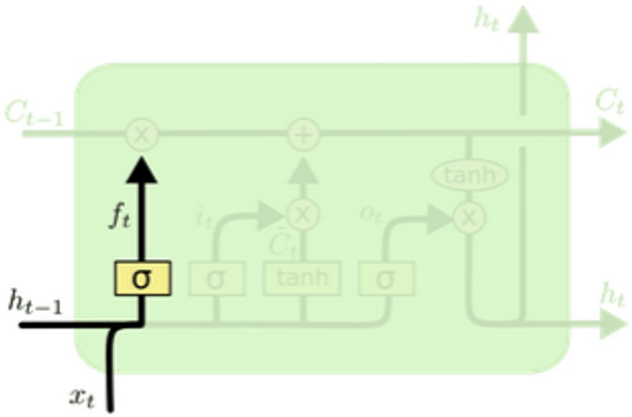
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

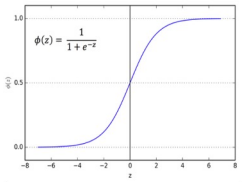
gate value

f_t = forget gate
Decides which information to delete that is not important from previous time step

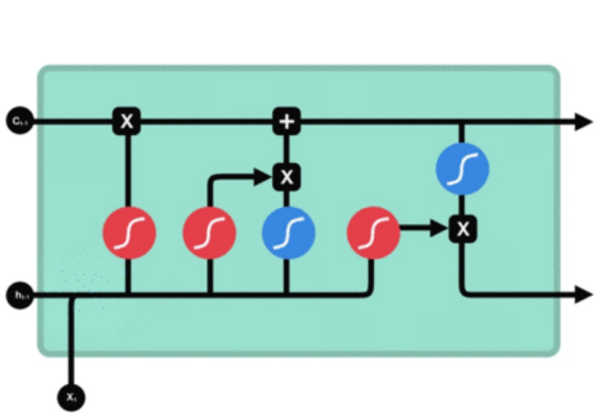
1. the current input x(t) + hidden state h(t-1) of the previous output —> [x(t), h(t-1)]

1. transform through a fully connected layer, and finally activate f(t) through the sigmoid function





Sigmoid Function —> (0,1)
Adjust the flow value.



Images from https://blog.csdn.net/mengxianglong123/article/details/126152251

LSTM-Input Gate

Decide How Much This Unit Adds to the Current State

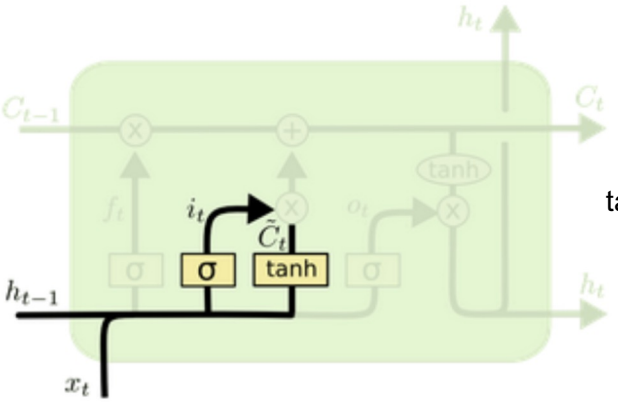
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

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

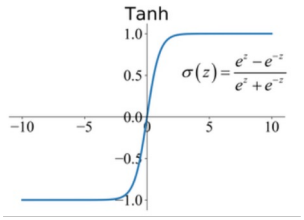
current cell state

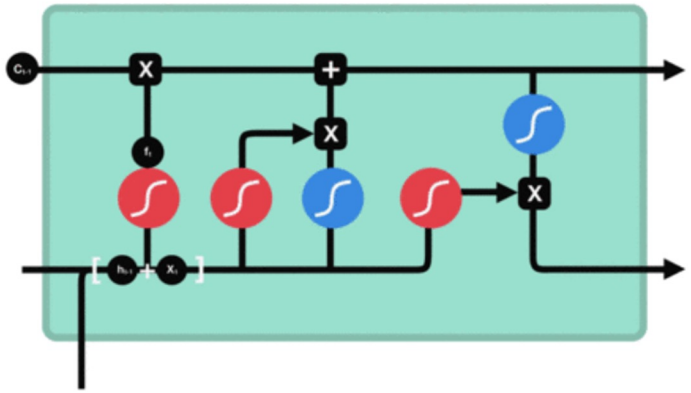
i_t = input gate
Determines which information to let through based on its significance in the current time step

Sigmoid function: it decides which values to let through (0 or 1).
tanh function: gives weightage to the values which are passed, deciding their level of importance (-1 to 1).



tanh Function —> (-1,1)

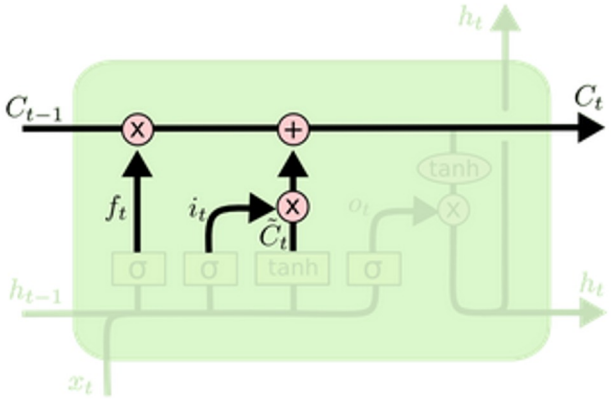


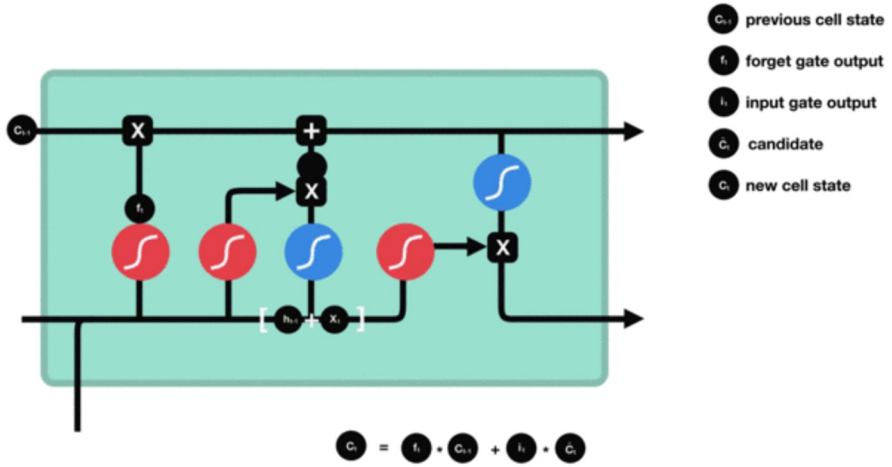


It gets the current cell state instead of the hidden state like the classic RNN.

Images from https://blog.csdn.net/mengxianglong123/article/details/126152251

LSTM Cell State





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

No fully connected layer here.

- Multiply forgotten gate value by the C(t-1) obtained in the previous time step
- Add Input gate value and C(t)

—> Updated C(t)

Cell State process is the application of input gate and forget gate.

Images from https://blog.csdn.net/mengxianglong123/article/details/126152251

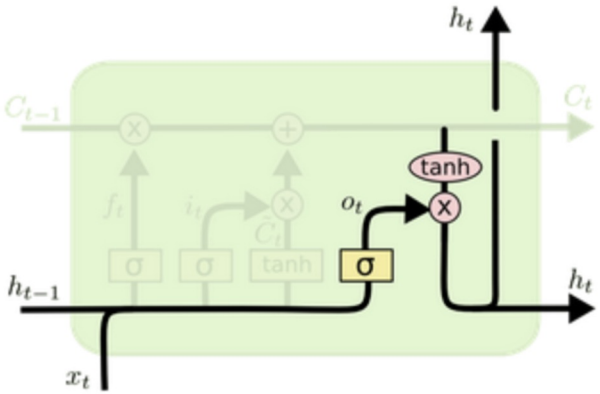
LSTM-Output Gate

Decide What Part of the Current Cell State Makes It to the Output

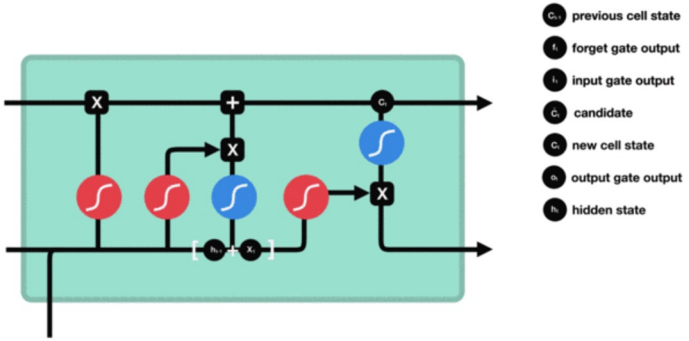
$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

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

o_t = output gate
Allows the passed in information to impact the output in the current time step



First, run a sigmoid layer to obtain the output gate value
Then, put the cell state through tanh to push the values to be between -1 and 1 and multiply it by the output of the sigmoid gate. —–> generate the hidden state h(t)



example: predict the next word in the sentence:

"John played tremendously well against the opponent and won for his team. For his contributions, brave ____ was awarded player of the match."
There could be many choices for the empty space.
The current input brave is an adjective, and adjectives describe a noun.
So, "John" could be the best output after brave.

LSTM

- **Advantages:**
  - The gate structure of LSTM can effectively slow down the gradient disappearance or explosion that may occur in long sequence problems. Although it cannot eliminate this phenomenon, it performs better than traditional RNN in longer sequence problems.
- **Disadvantages:**
  - Due to the relatively complex internal structure, the training efficiency is much lower than that of traditional RNN under the same computing power.

Application

- **Sequence-to-sequence (Seq2Seq) tasks**: machine translation, speech-to-text, and text-to-speech conversion.
- **Sequence labeling tasks**: entity recognition, part-of-speech tagging, and sentiment analysis.
- **Sequence classification tasks**: text classification, emotion recognition from speech, and gesture recognition from video sequences.
- **Sequence generation tasks:** text generation, music composition, and image captioning.

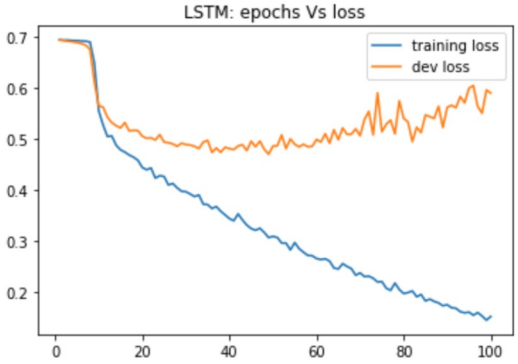
Implementation

Sentiment analysis by movie comments

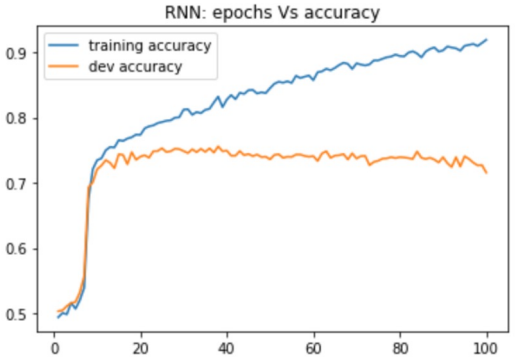
- Model: RNN & LSTM
- Dataset: IMDB Movie Reviews Dataset
- Development tool: Python 3.6
- Training Epoch: 100
- Used Libraries: PyTorch

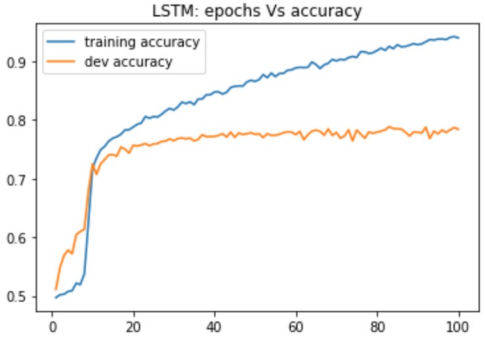
49582 unique values	2 unique values
A wonderful little production. The filming technique is very unassuming- very old-time-B...	positive
I thought this was a wonderful way to spend time on a too hot summer weekend, sitting in the air con...	positive

Result



	precision	recall	f1-score
0	0.70	0.74	0.72
1	0.72	0.69	0.70





	precision	recall	f1-score
0	0.77	0.79	0.78
1	0.78	0.77	0.78

Open Issue

- **Long-Term Dependencies**: Even with LSTMs, learning long-term dependencies can still be challenging.
- **Training Time and Computational Resources**: Training RNNs, especially on long sequences, can be time-consuming and computationally intensive.
- **Difficulty in Parallelization**: Unlike feedforward networks and convolutional neural networks, the inherently sequential computation in RNNs makes it hard to fully utilize the modern computational resources like GPUs, resulting in slower training times.
- **Interpretability**: Like many other deep learning models, RNNs are often criticized for being "black boxes".
- **Bi-Directional Context**: Standard RNNs can struggle with contexts that should logically come from "future" data points.
- **Optimization**: Training RNNs is a difficult optimization problem, due to the non-convex nature of the loss surface.
- **Overfitting**: Like other neural networks, RNNs can also overfit to the training data, especially when the model complexity (number of parameters) is high relative to the amount of training data.

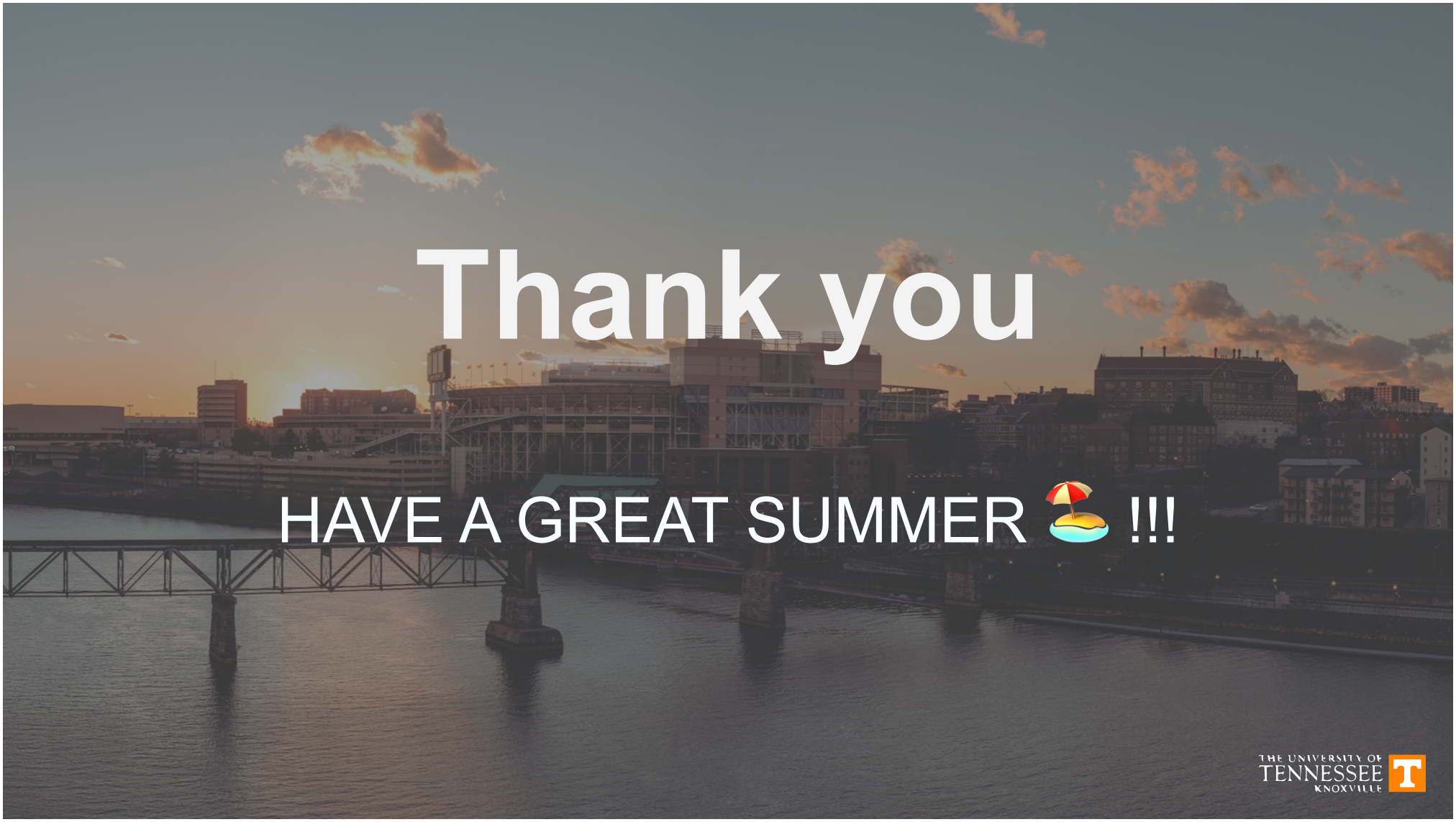
Reference

1. Applications of Recurrent Neural Networks (RNNs) https://iq.opengenus.org/applications-of-rnn/
2. Movie Review Sentiment Analysis LSTM https://www.kaggle.com/code/sanket30/movie-review-sentiment-analysis-lstm
3. A Method Of Emotional Analysis Of Movie Based On Convolution Neural Network And Bi-directional LSTM RNN
4. Recurrent neural network https://en.wikipedia.org/wiki/Recurrent_neural_network#cite_note-14
5. Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network
6. Illustrated Guide to Recurrent Neural Networks https://towardsdatascience.com/illustrated-guide-to-recurrent-neural-networks-79e5eb8049c9
7. Image from https://www.simplilearn.com/tutorials/deep-learning-tutorial/rnn
8. https://medium.com/swlh/simple-explanation-of-recurrent-neural-network-rnn-1285749cc363
9. https://blog.csdn.net/mengxianglong123/article/details/126152251
10. A Hybrid CNN-LSTM Model for Improving Accuracy of Movie Reviews Sentiment Analysis https://link.springer.com/article/10.1007/s11042-019-07788-7

Test Questions Revisit

- Type of RNNs and one of their application.
- The main issue of vanilla RNN and raise one solution.
- List main mechanisms in LSTM (four).


Questions?
THE UNIVERSITY OF
TENNESSEE
KNOXVILLE


Thank you
HAVE A GREAT SUMMER 🏝 !!!
THE UNIVERSITY OF TENNESSEE KNOXVILLE