PATTERN RECOGNITION USING PYTHON

Recurrent Neural Networks

Wen-Yen Hsu

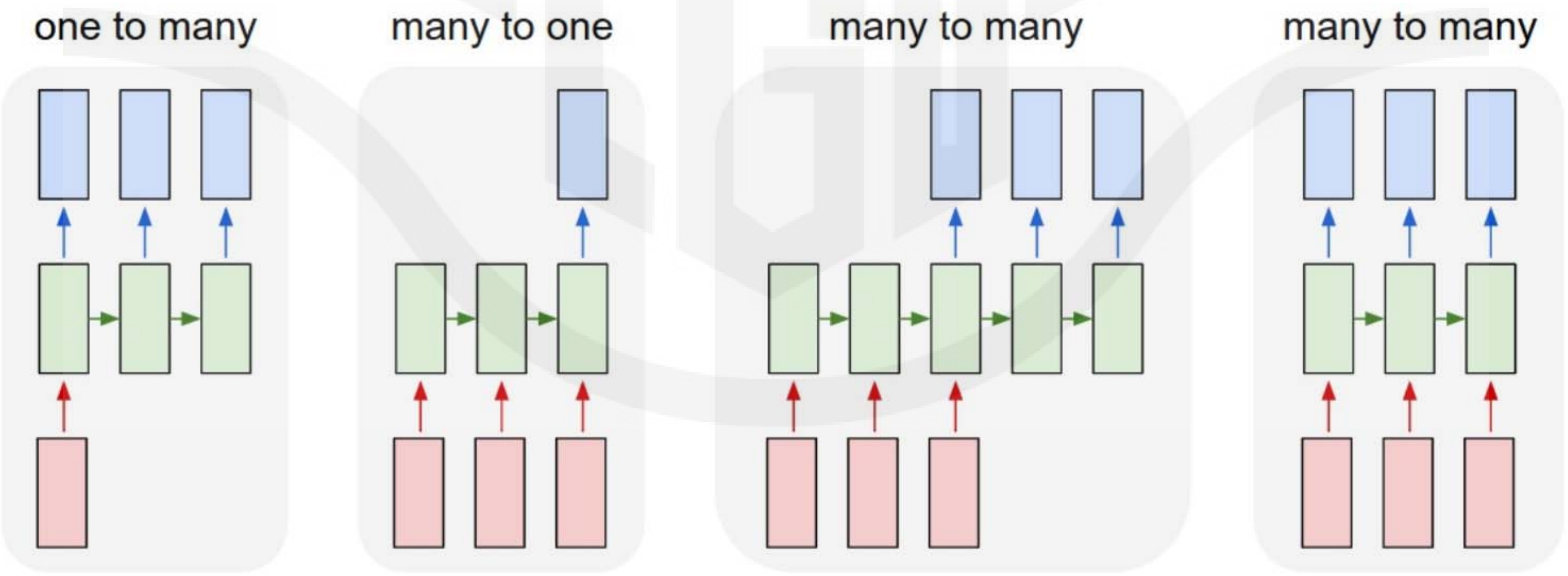
Dept Electrical Engineering

Chang Gung University, Taiwan

2019-Spring

Serial Data Format (Sequence Model)

- One to Many (image captioning)
- Many to One (sentiment analysis)
- Many to Many (machine translating)
- Many to Many (video description, character-level language model)



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

Sequence Model Application

- One to Many (image captioning)

A person riding a motorcycle on a dirt road.

Two dogs play in the grass.

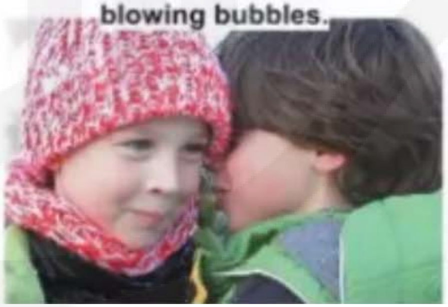
A skateboarder does a trick on a ramp.

A dog is jumping to catch a frisbee.

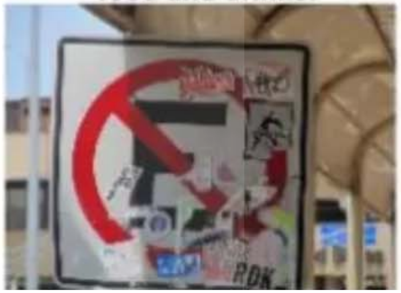
A group of young people playing a game of frisbee.

Two hockey players are fighting over the puck.

A little girl in a pink hat is blowing bubbles.

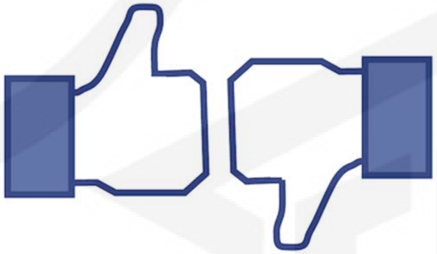


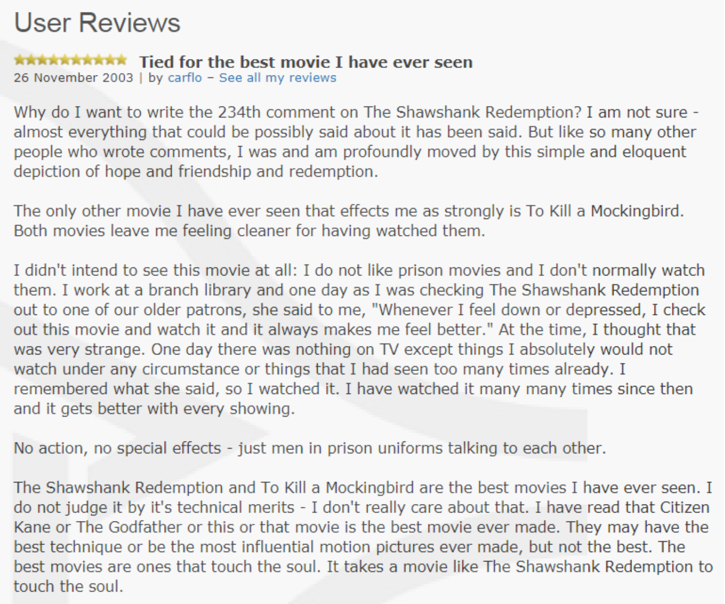
A refrigerator filled with lots of food and drinks.



Sequence Model Application (Cont.)

- Many to One (sentiment analysis)





User Reviews

★★★★★★★★★★ **Tied for the best movie I have ever seen**
26 November 2003 | by carflo – See all my reviews

Why do I want to write the 234th comment on The Shawshank Redemption? I am not sure - almost everything that could be possibly said about it has been said. But like so many other people who wrote comments, I was and am profoundly moved by this simple and eloquent depiction of hope and friendship and redemption.

The only other movie I have ever seen that effects me as strongly is To Kill a Mockingbird. Both movies leave me feeling cleaner for having watched them.

I didn't intend to see this movie at all: I do not like prison movies and I don't normally watch them. I work at a branch library and one day as I was checking The Shawshank Redemption out to one of our older patrons, she said to me, "Whenever I feel down or depressed, I check out this movie and watch it and it always makes me feel better." At the time, I thought that was very strange. One day there was nothing on TV except things I absolutely would not watch under any circumstance or things that I had seen too many times already. I remembered what she said, so I watched it. I have watched it many many times since then and it gets better with every showing.

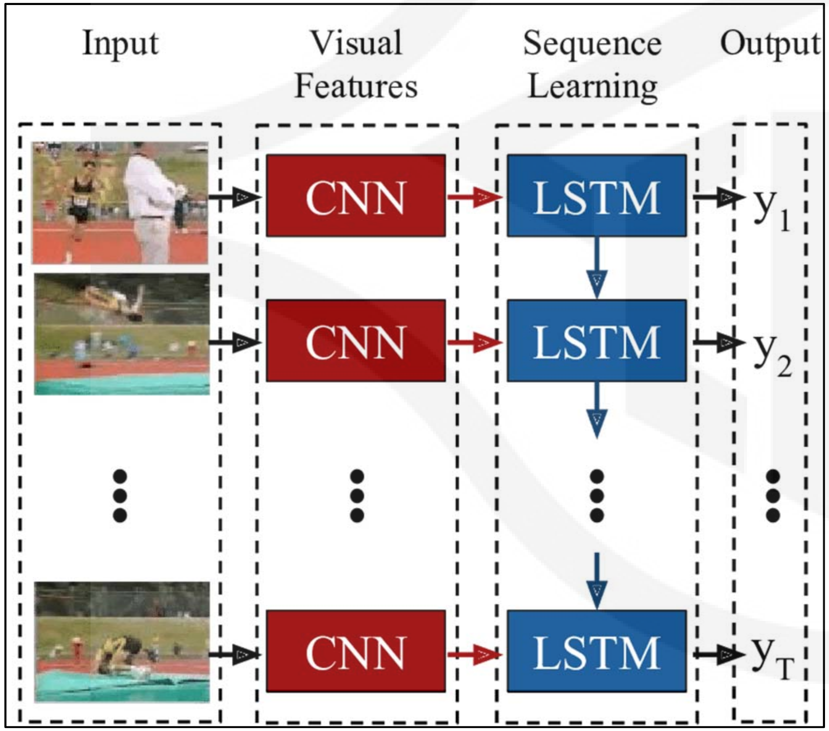
No action, no special effects - just men in prison uniforms talking to each other.

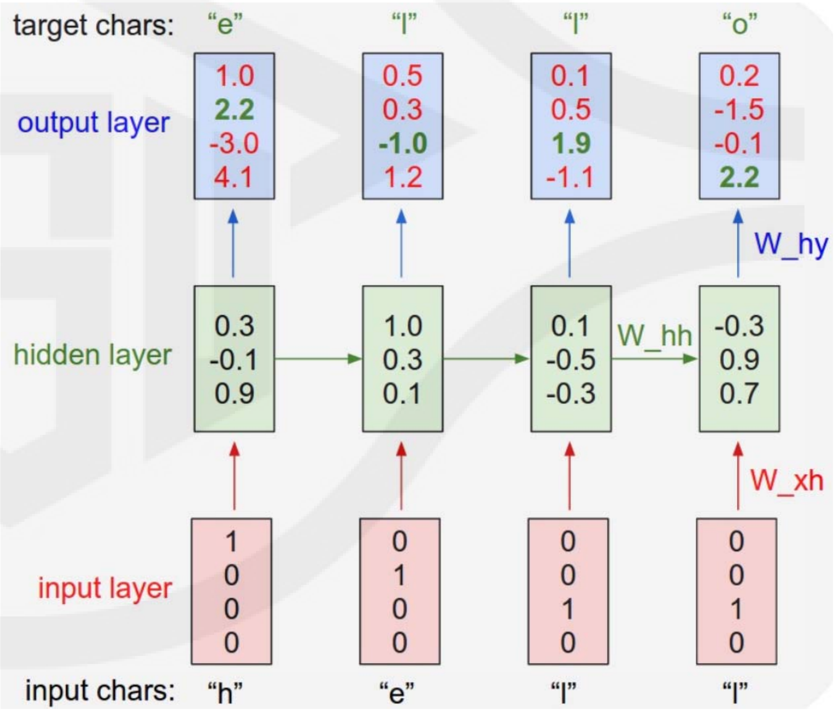
The Shawshank Redemption and To Kill a Mockingbird are the best movies I have ever seen. I do not judge it by it's technical merits - I don't really care about that. I have read that Citizen Kane or The Godfather or this or that movie is the best movie ever made. They may have the best technique or be the most influential motion pictures ever made, but not the best. The best movies are ones that touch the soul. It takes a movie like The Shawshank Redemption to touch the soul.

- Many to Many (machine translating)

Sequence Model Application (Cont.)

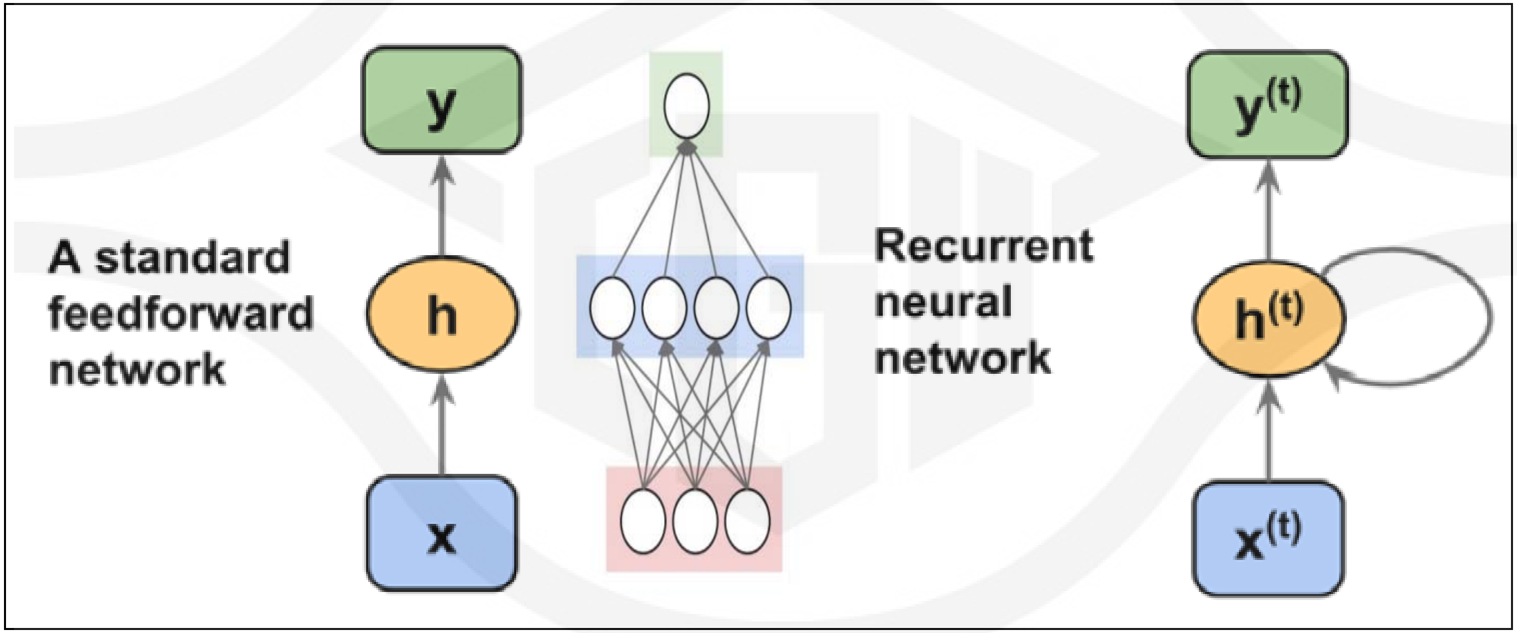
- Many to Many (video description , character-level language model)





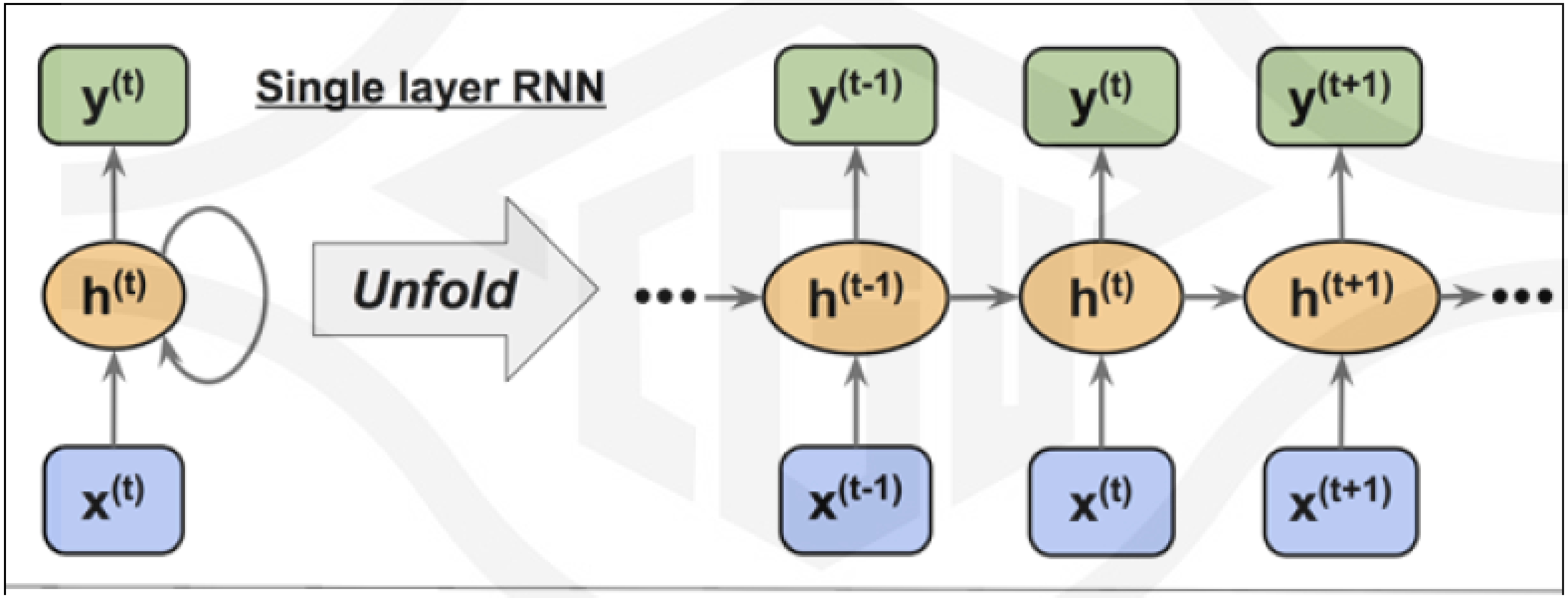
Recurrent Neural Networks

- Recurrent Neural Network : feed forward network have a **recurrent edge** at hidden unit



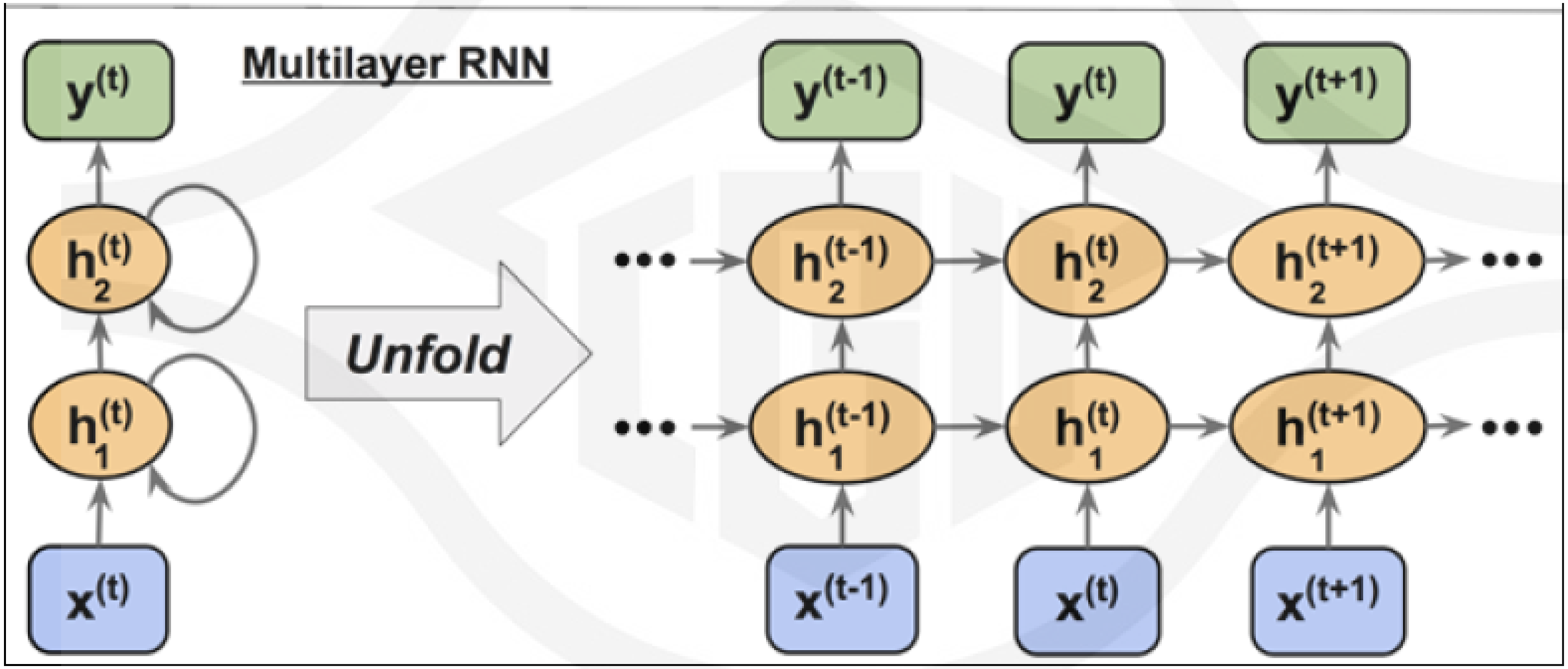
Recurrent Neural Networks Structure

- Unfolded **recurrent edge**



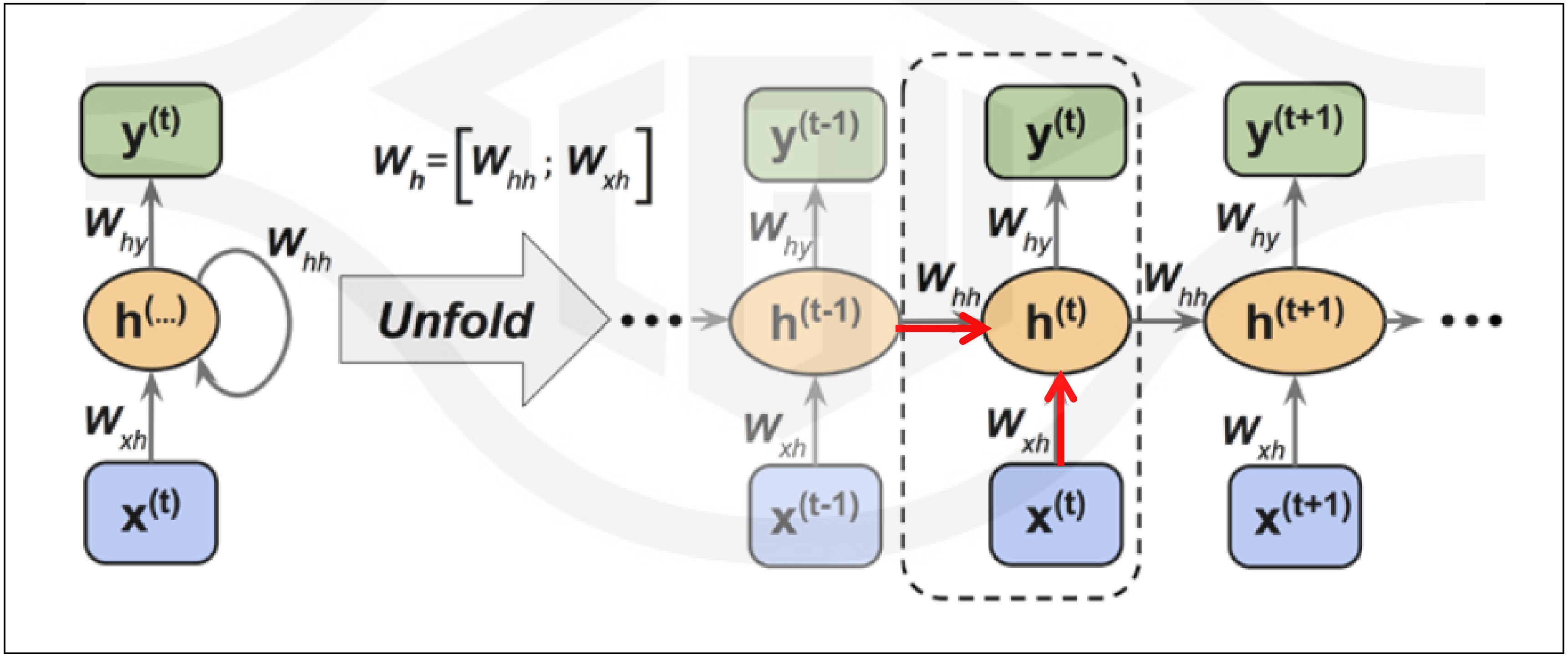
Multi Layer Recurrent Neural Networks

- Have two hidden layer or more

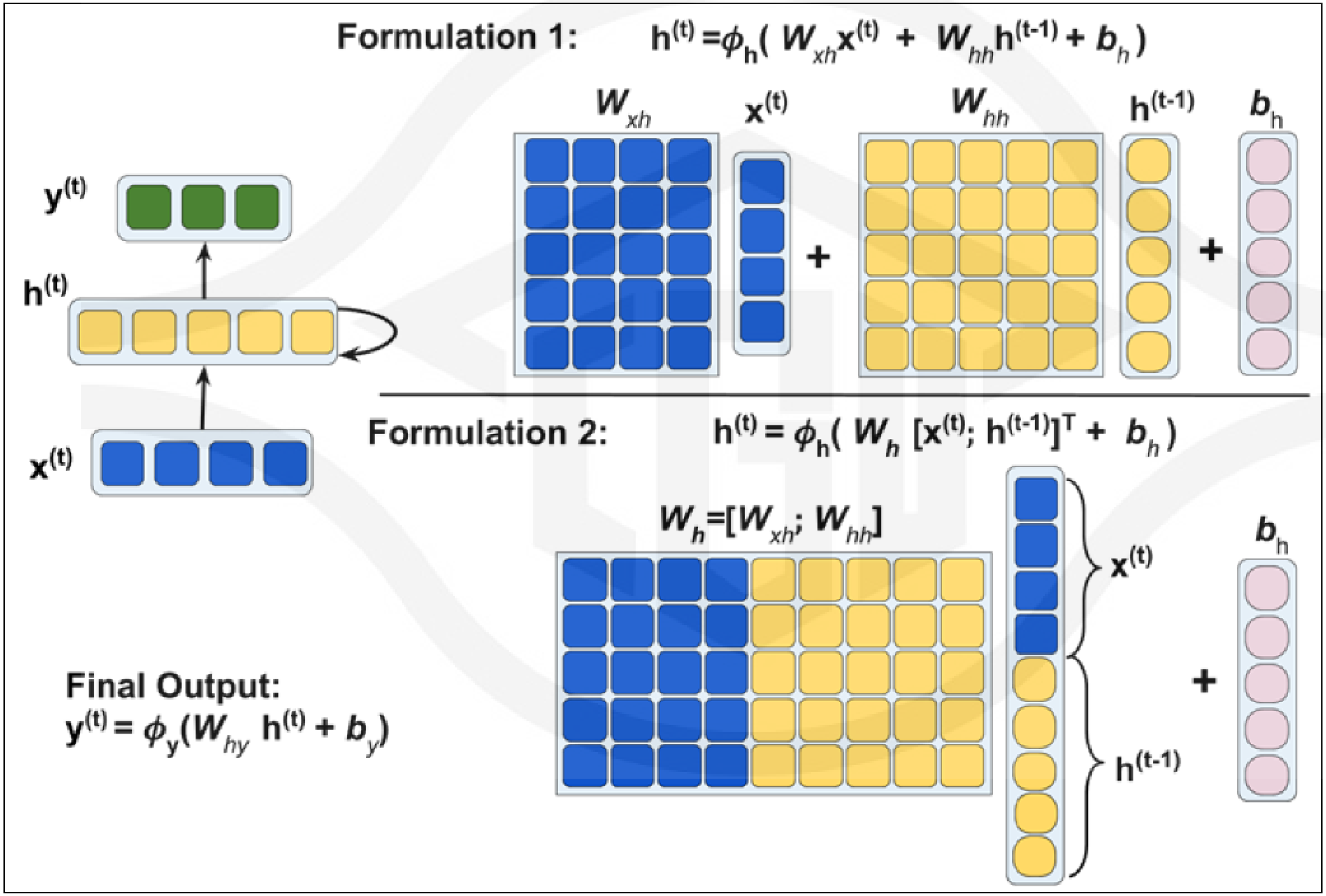


Computing Flow

- Hidden unit receives two distinct sets of input
  - From the input layer
  - Same hidden layer from the previous time step t-1



Weight Matrix


Formulation 1: $h^{(t)} = \phi_h(W_{xh} x^{(t)} + W_{hh} h^{(t-1)} + b_h)$
W_{xh}
$x^{(t)}$
W_{hh}
$h^{(t-1)}$
b_h
+
+
$y^{(t)}$
$h^{(t)}$
$x^{(t)}$
Formulation 2: $h^{(t)} = \phi_h(W_h \, [x^{(t)}; h^{(t-1)}]^T + b_h)$
$W_h = [W_{xh}; W_{hh}]$
$x^{(t)}$
$h^{(t-1)}$
b_h
+
Final Output:
$y^{(t)} = \phi_y(W_{hy} \, h^{(t)} + b_y)$

Loss Function

- Overall loss *L* is the sum of all the loss functions at times *t* =1 to *t* = *T* :

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

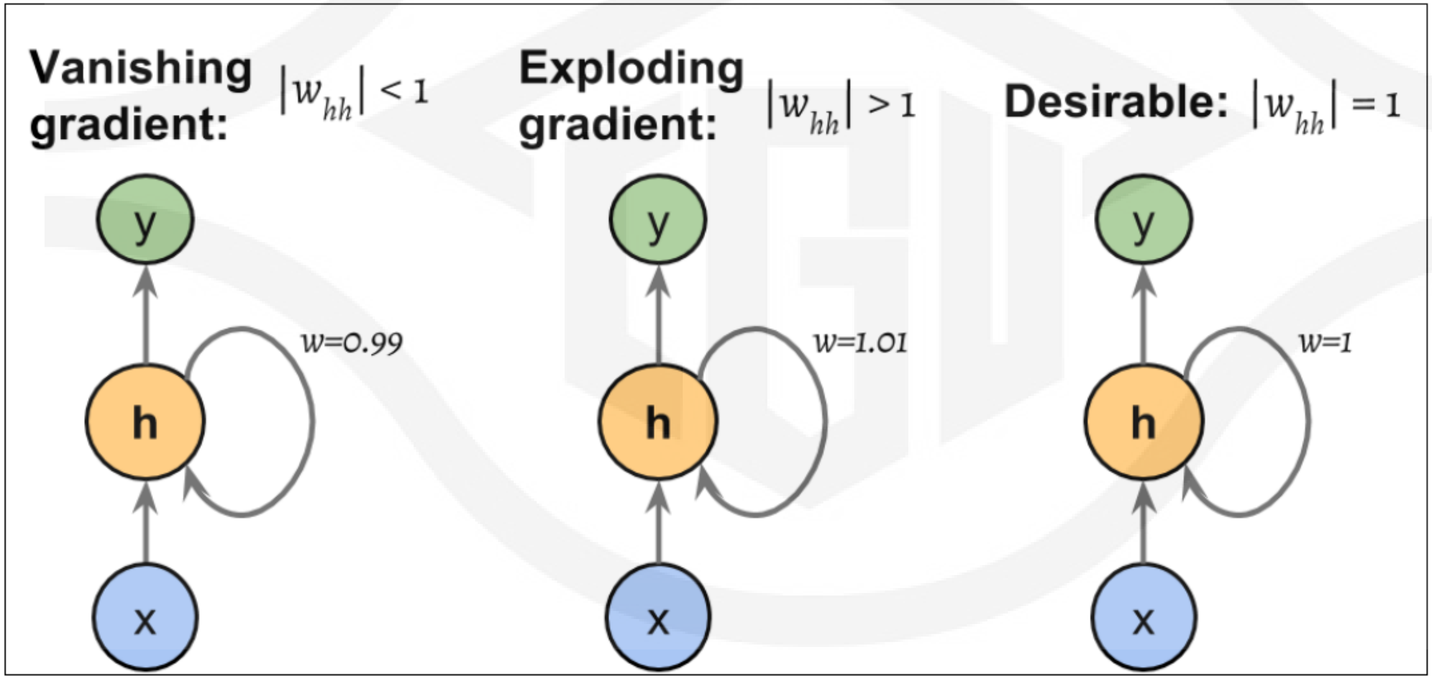
- The gradient will be computed as follows

$$\frac{\partial L^{(t)}}{\partial W_{hh}} = \frac{\partial L^{(t)}}{\partial \boldsymbol{y}^{(t)}} \times \frac{\partial \boldsymbol{y}^{(t)}}{\partial \boldsymbol{h}^{(t)}} \times \left(\sum_{k=1}^{t} \frac{\partial \boldsymbol{h}^{(t)}}{\partial \boldsymbol{h}^{(k)}} \times \frac{\partial \boldsymbol{h}^{(k)}}{\partial W_{hh}} \right)$$

$$\frac{\partial \boldsymbol{h}^{(t)}}{\partial \boldsymbol{h}^{(k)}} = \prod_{i=k+1}^{t} \frac{\partial \boldsymbol{h}^{(i)}}{\partial \boldsymbol{h}^{(i-1)}}$$

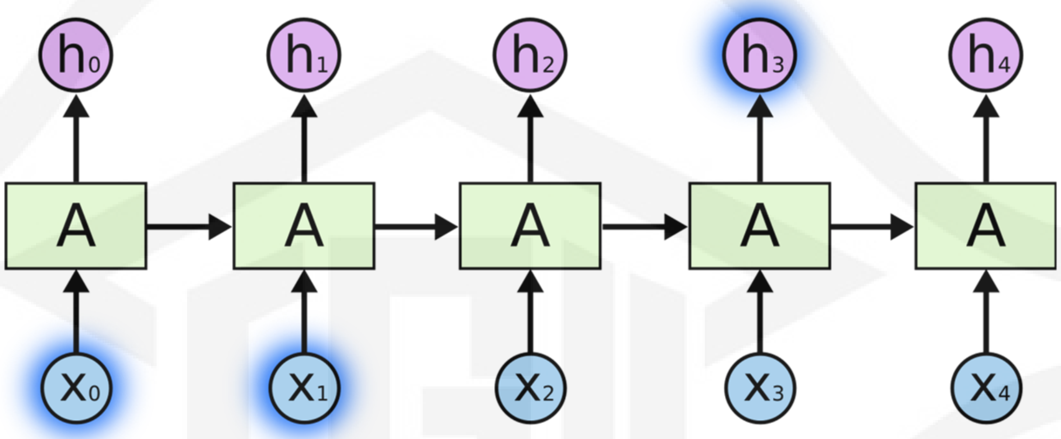
Gradient Vanishing or Exploding

- $\dfrac{\partial \boldsymbol{h}^{(t)}}{\partial \boldsymbol{h}^{(k)}}$ has $\boldsymbol{t-k}$ multiplications, multiplying the w weight $\boldsymbol{t-k}$ times results in a factor w^{t-k}.

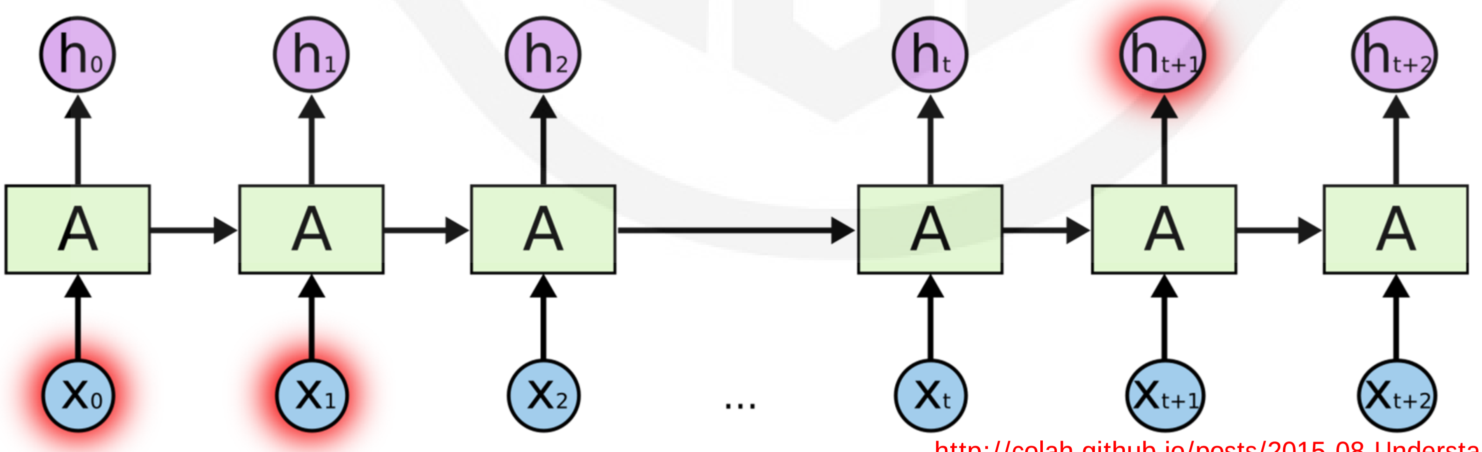

Vanishing gradient: $|w_{hh}| < 1$
Exploding gradient: $|w_{hh}| > 1$
Desirable: $|w_{hh}| = 1$
y
h
x
w=0.99
y
h
x
w=1.01
y
h
x
w=1

Why LSTM ? (long-term dependency problem)

- Short-term dependency



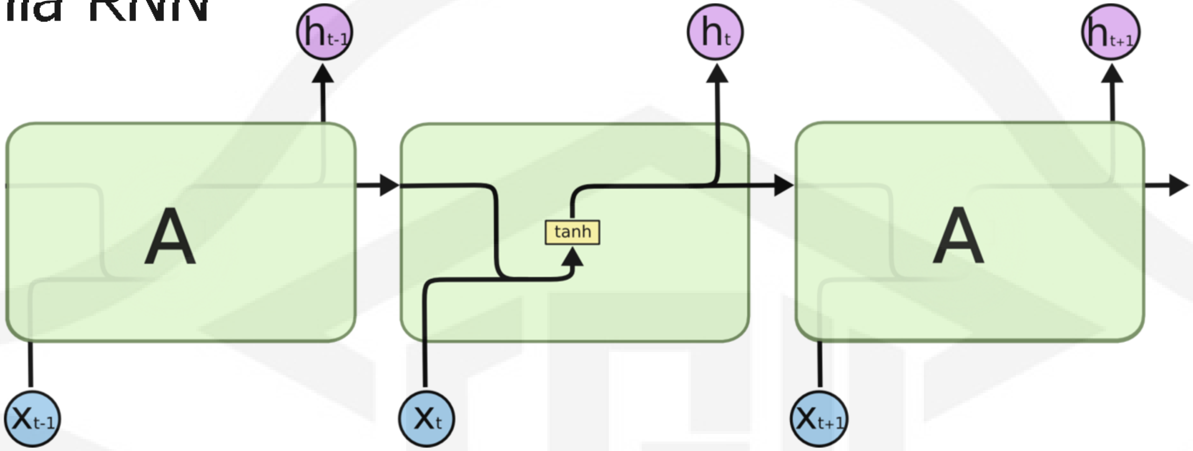
- Long-term dependency



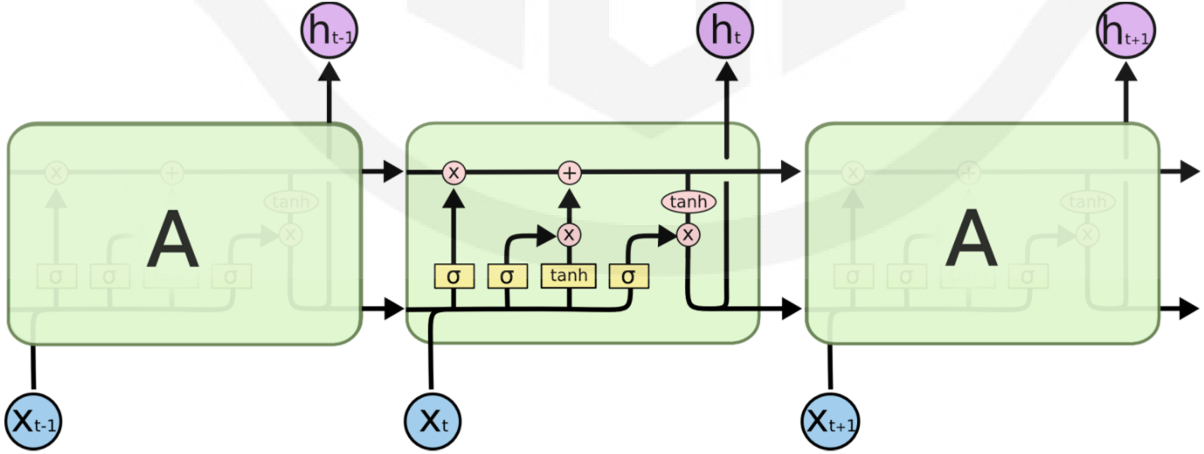
http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long Short-Term Memory (Hochreiter et al., 1997)

- Vanilla RNN



- LSTM Cell



http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long Short-Term Memory Cell

- Cell Output
  - Cell state (***C****t*)
  - Hidden state (***h****t*)
- Cell Control
  - Forget gate (***f****t*)
  - Input gate (***i****t*)
  - Output gate (***o****t*)

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

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

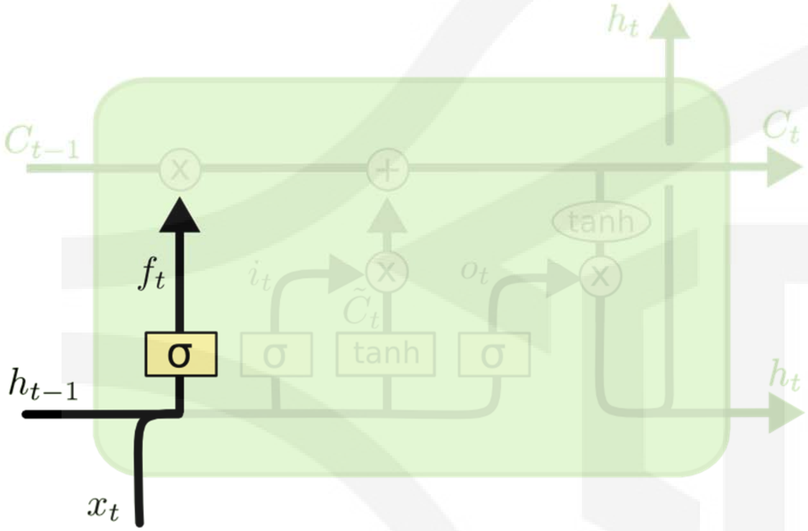
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

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

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

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

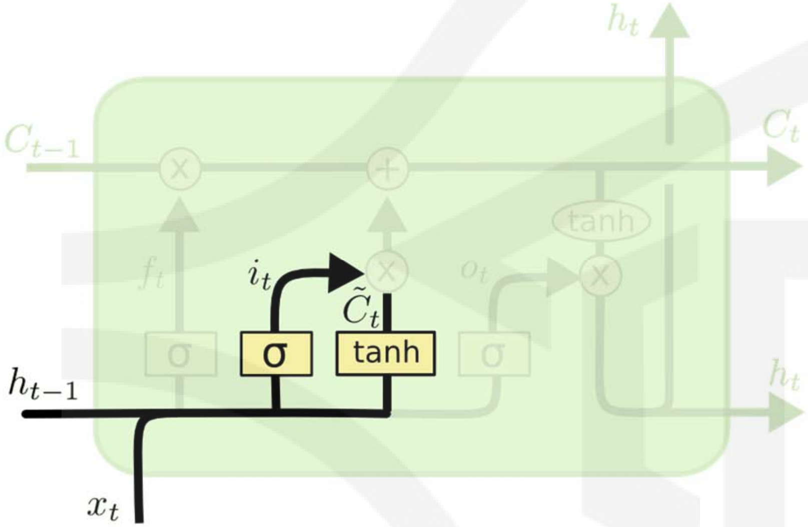
Forget gate



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

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Input gate

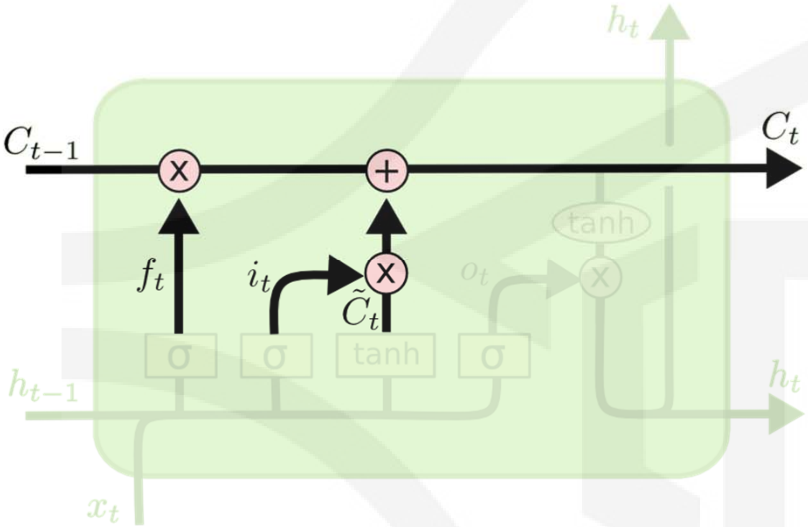


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

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

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

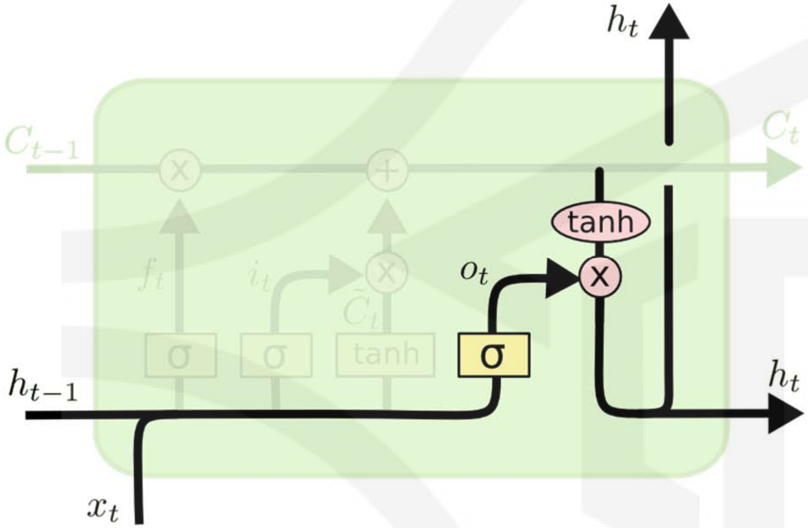
Cell state



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

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Output gate Hidden state

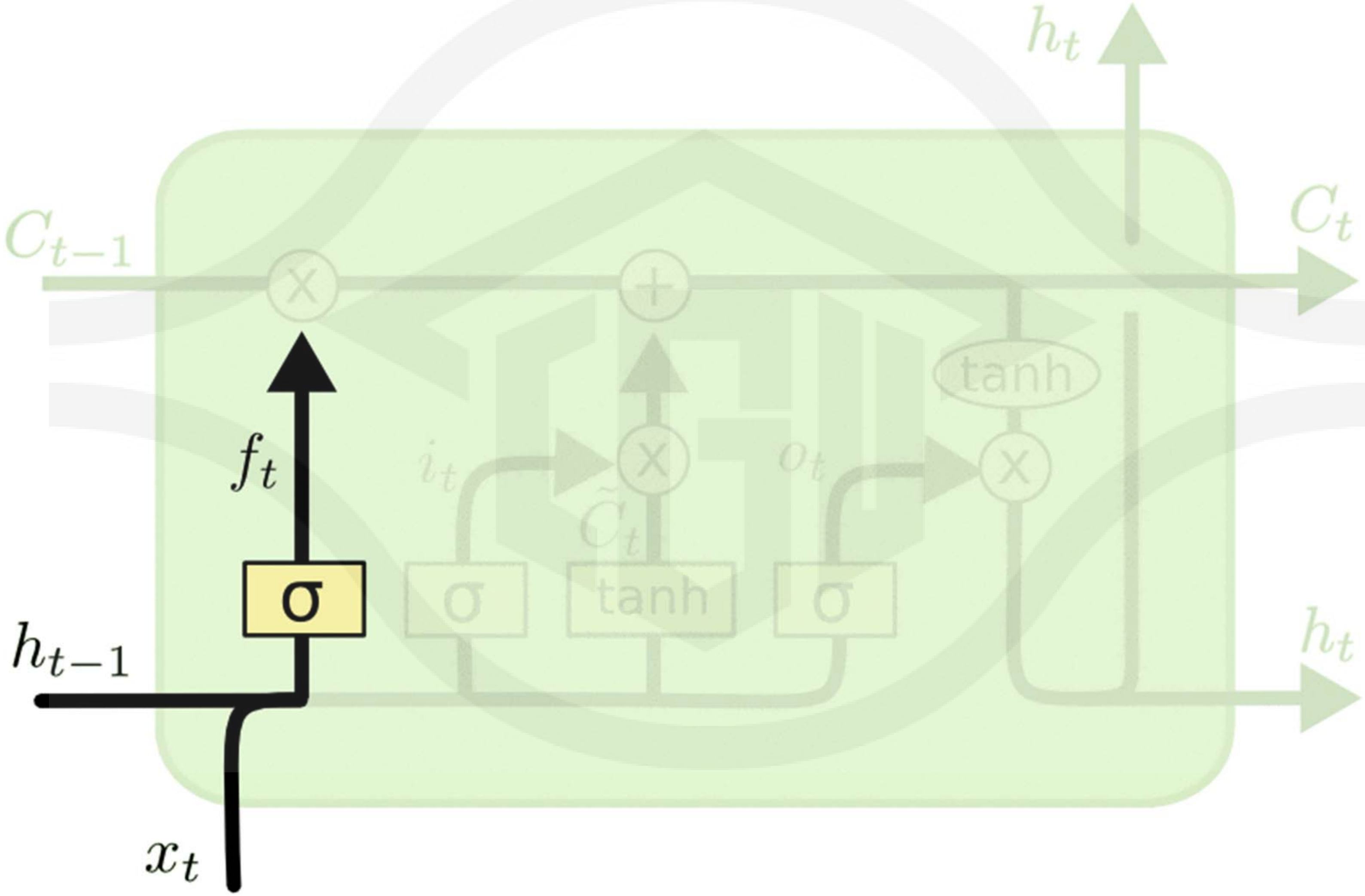


$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$

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

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

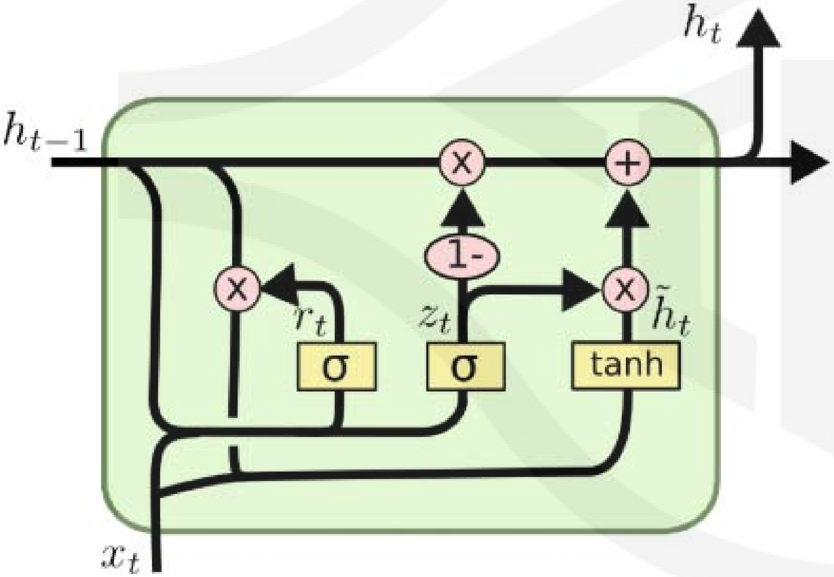
Long Short-Term Memory Structure



http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Gated Recurrent Unit (Cho et al., 2014)

- Simplified LSTM
- Combine forget gate and input gate into update gate



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

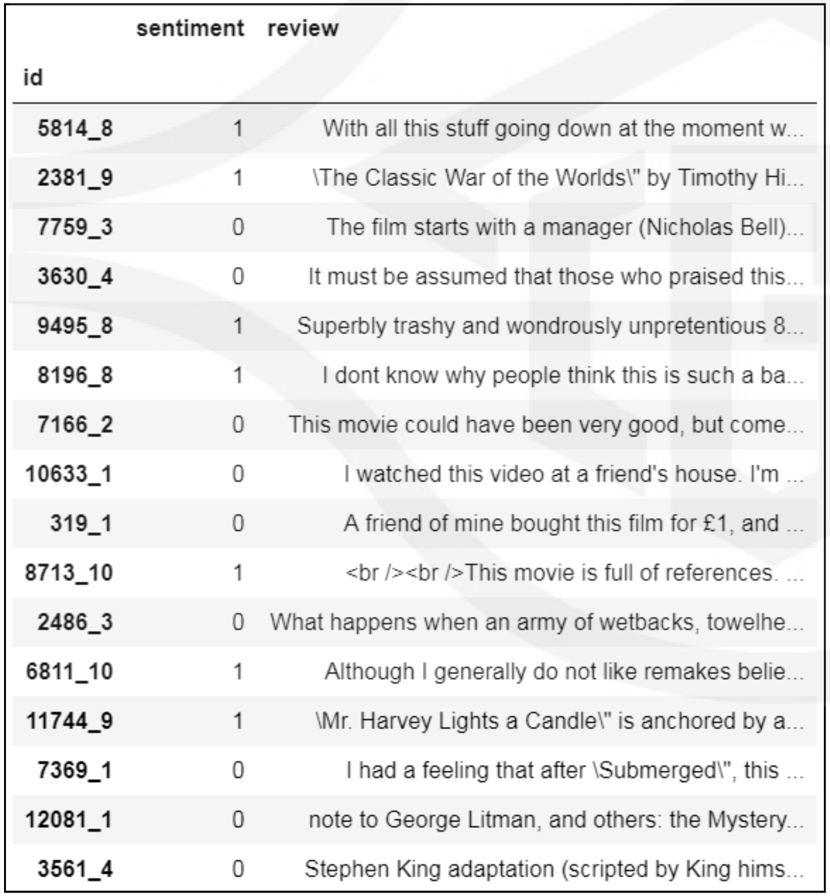
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

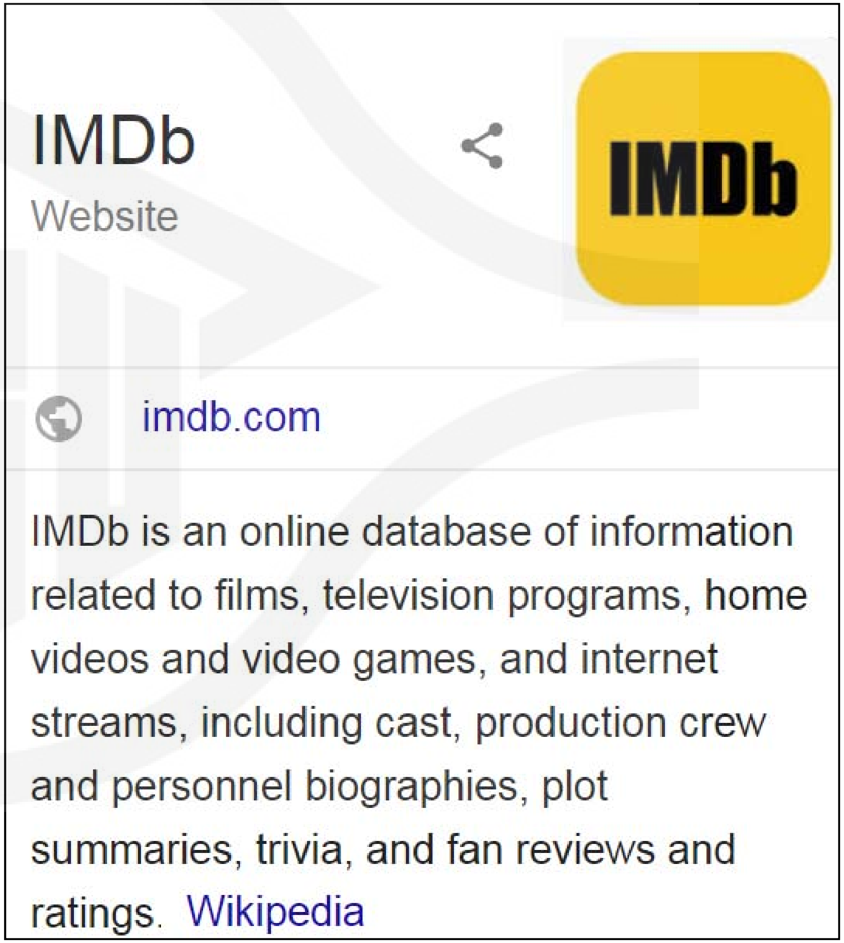
http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Sentiment Analysis of IMDb Movie Reviews Data

- 25,000 reviews with labels 0/1 (dislike/like)
- 25,000 reviews without labels



id	sentiment	review
5814_8	1	With all this stuff going down at the moment w...
2381_9	1	\The Classic War of the Worlds\" by Timothy Hi...
7759_3	0	The film starts with a manager (Nicholas Bell)...
3630_4	0	It must be assumed that those who praised this...
9495_8	1	Superbly trashy and wondrously unpretentious 8...
8196_8	1	I dont know why people think this is such a ba...
7166_2	0	This movie could have been very good, but come...
10633_1	0	I watched this video at a friend's house. I'm ...
319_1	0	A friend of mine bought this film for £1, and ...
8713_10	1	<br /><br />This movie is full of references. ...
2486_3	0	What happens when an army of wetbacks, towelhe...
6811_10	1	Although I generally do not like remakes belie...
11744_9	1	\Mr. Harvey Lights a Candle\" is anchored by a...
7369_1	0	I had a feeling that after \Submerged\", this ...
12081_1	0	note to George Litman, and others: the Mystery...
3561_4	0	Stephen King adaptation (scripted by King hims...



IMDb

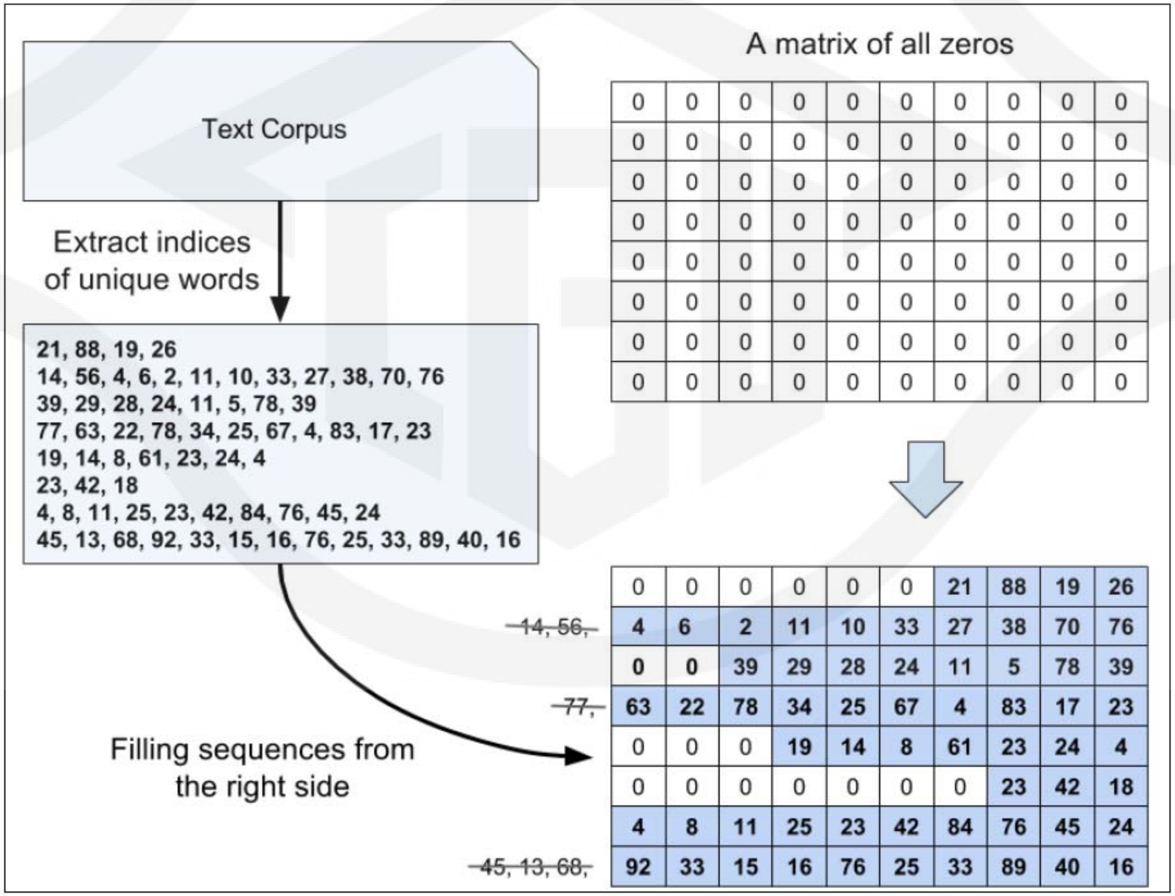
Website

imdb.com

IMDb is an online database of information related to films, television programs, home videos and video games, and internet streams, including cast, production crew and personnel biographies, plot summaries, trivia, and fan reviews and ratings. Wikipedia

Preparing the Data

- Arrange text to a suitable format for many to one RNN structure



Text Corpus

Extract indices of unique words

21, 88, 19, 26
14, 56, 4, 6, 2, 11, 10, 33, 27, 38, 70, 76
39, 29, 28, 24, 11, 5, 78, 39
77, 63, 22, 78, 34, 25, 67, 4, 83, 17, 23
19, 14, 8, 61, 23, 24, 4
23, 42, 18
4, 8, 11, 25, 23, 42, 84, 76, 45, 24
45, 13, 68, 92, 33, 15, 16, 76, 25, 33, 89, 40, 16

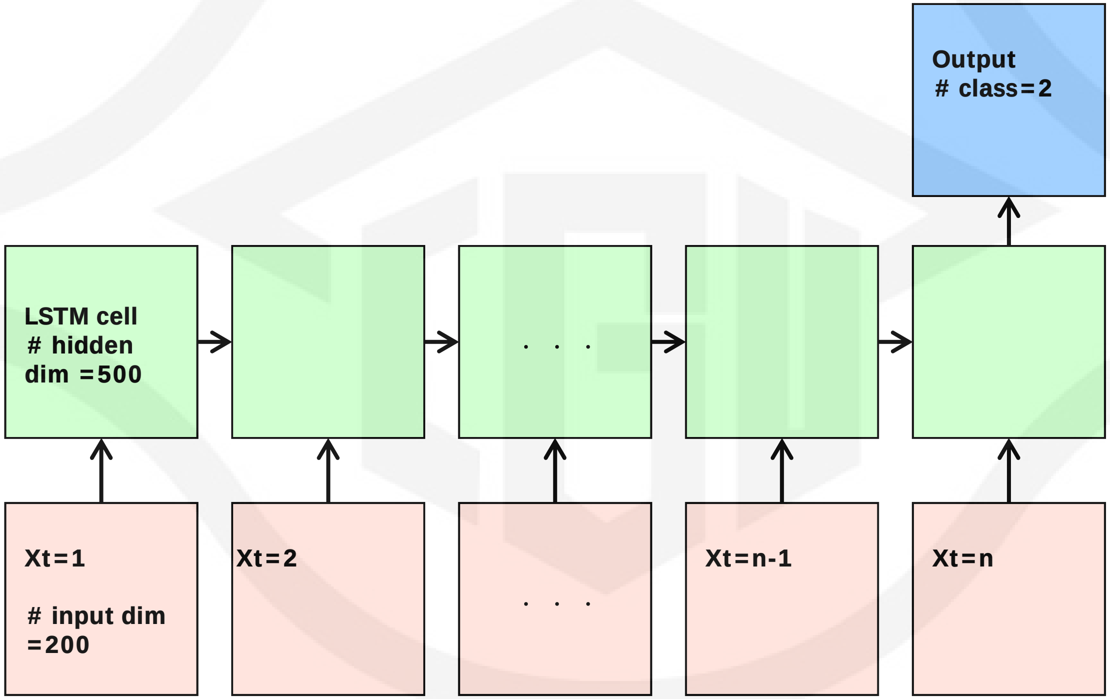
A matrix of all zeros

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Filling sequences from the right side

	0	0	0	0	0	0	21	88	19	26
~~14, 56,~~	4	6	2	11	10	33	27	38	70	76
	0	0	39	29	28	24	11	5	78	39
~~77,~~	63	22	78	34	25	67	4	83	17	23
	0	0	0	19	14	8	61	23	24	4
	0	0	0	0	0	0	0	23	42	18
	4	8	11	25	23	42	84	76	45	24
~~45, 13, 68,~~	92	33	15	16	76	25	33	89	40	16

Many to One (sentiment analysis) RNN

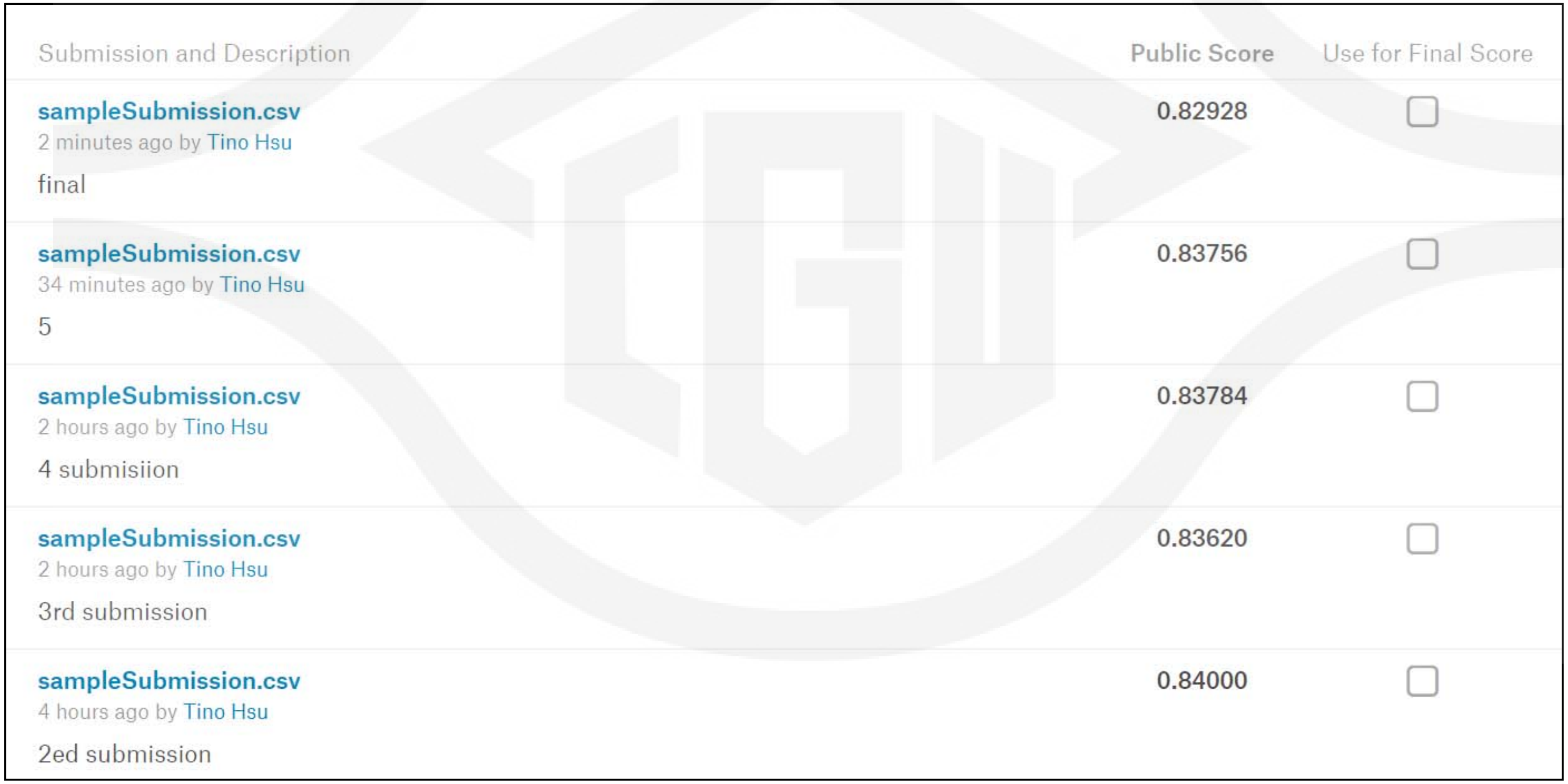


RNN with PyTorch

```python
class RNN(nn.Module):
    def __init__(self):
        super(RNN, self).__init__()
        self.rnn = nn.LSTM(
            input_size=embed_dim,
            hidden_size=500,
            num_layers=1,
            batch_first=True,
            # True (batch, time_step, input_size)
        )
        nn.init.orthogonal_(self.rnn.weight_ih_l0)
        nn.init.orthogonal_(self.rnn.weight_hh_l0)
        nn.init.zeros_(self.rnn.bias_ih_l0)
        nn.init.zeros_(self.rnn.bias_hh_l0)
        self.fc = nn.Linear(500, 2)
    def forward(self, x):
        # x shape (batch, time_step, input_size)
        rnn_out, (hn, cn) = self.rnn(x)
        # r_out shape (batch, time_step, output_size)
        out = self.fc(rnn_out[:, -1, :])
        # Pick the the last time of r_out output
        return out
```

Kaggle Leagerboard

- https://www.kaggle.com/c/word2vec-nlp-tutorial/leaderboard



Submission and Description	Public Score	Use for Final Score
sampleSubmission.csv 2 minutes ago by Tino Hsu final	0.82928	□
sampleSubmission.csv 34 minutes ago by Tino Hsu 5	0.83756	□
sampleSubmission.csv 2 hours ago by Tino Hsu 4 submisiion	0.83784	□
sampleSubmission.csv 2 hours ago by Tino Hsu 3rd submission	0.83620	□
sampleSubmission.csv 4 hours ago by Tino Hsu 2ed submission	0.84000	□

Reference

- Sebastian Raschka, Vahid Mirjalili. Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow. Second Edition. Packt Publishing, 2017.
- Vishnu Subramanian. Deep Learning with PyTorch: A practical approach to building neural network models using PyTorch. Packt Publishing, 2018.