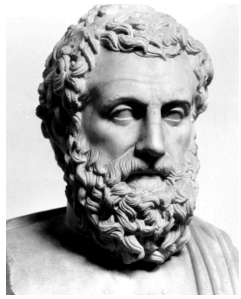
Artificial Intelligence For NLP Lesson - 13

人工智能与自然语言处理课程组

2019.Sept. 28



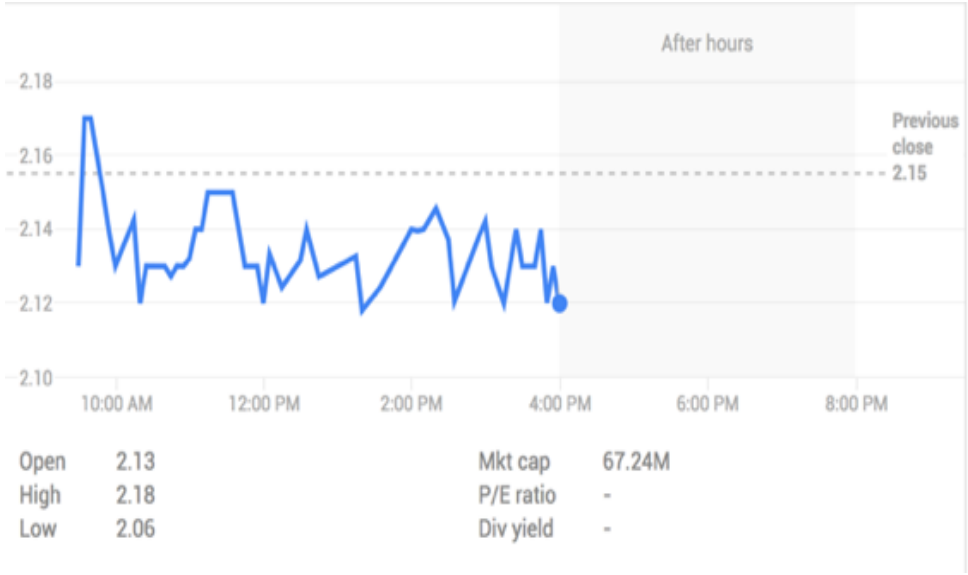
Review

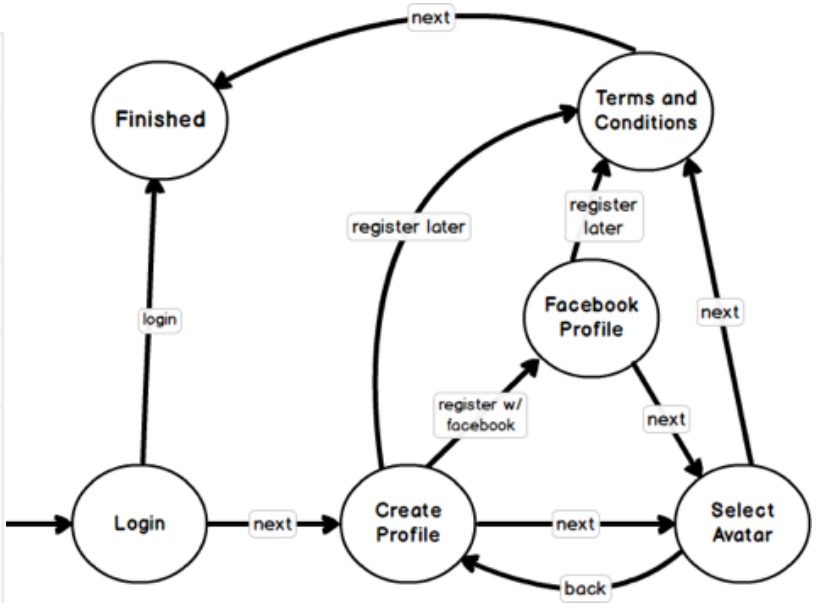
- Neural Networks
- Word Embedding
- Convolutional Neural Networks

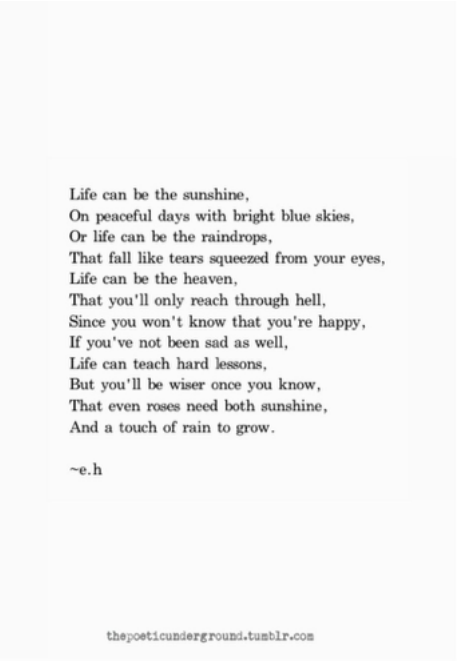
Outline

- RNN Background
- LSTM, GRU
- RNN in pure python
- RNN in Tensorflow
- RNN in Keras
- Some Useful applications for RNN

- Stocks, Text, User Actions





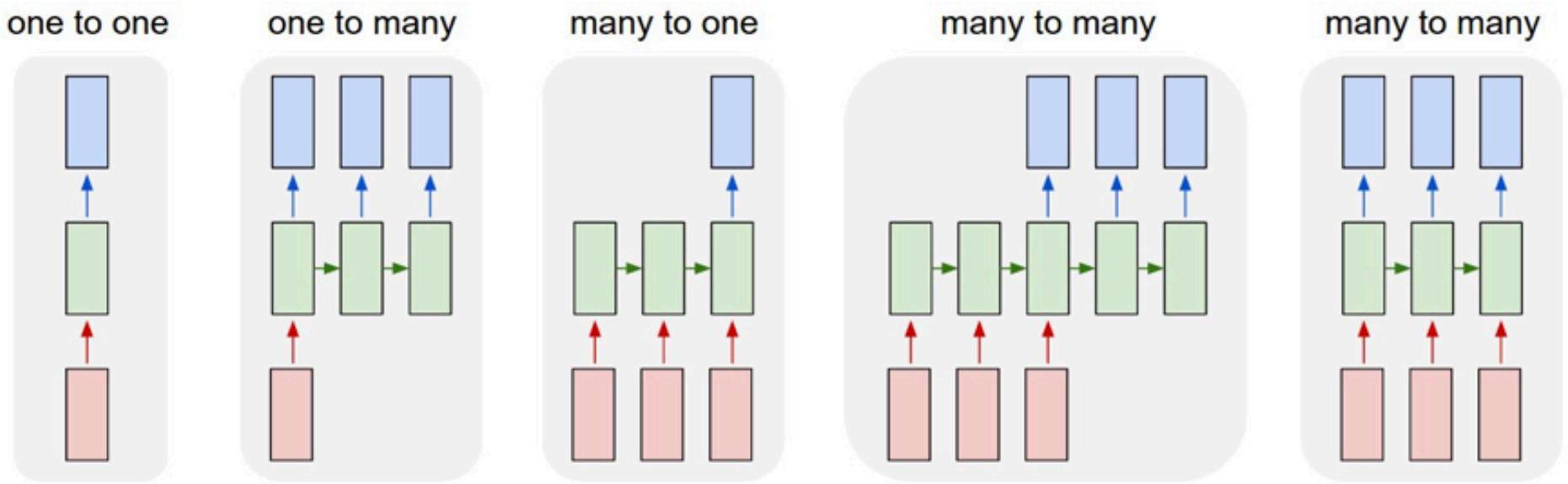

Life can be the sunshine,
On peaceful days with bright blue skies,
Or life can be the raindrops,
That fall like tears squeezed from your eyes,
Life can be the heaven,
That you'll only reach through hell,
Since you won't know that you're happy,
If you've not been sad as well,
Life can teach hard lessons,
But you'll be wiser once you know,
That even roses need both sunshine,
And a touch of rain to grow.

~e.h

thepoeticunderground.tumblr.com

Sequence Problems

- States Dependency: The current state is based on the previous states.
- Variable Input Length: The Input length may change over cases.



In [3]:
```
def fib(n):
    if n == 0 or n == 1: return 1
    else:
        return fib(n-1) + fib(n - 2)
```

In [14]:
```
def fac(n):
    if n == 0: return 1
    else: return n * fac(n-1)
```

In [18]:
```
for i in range(10): print("{}\t{}".format(fib(i), fac(i)))
```

```
1	1
1	1
2	2
3	6
5	24
8	120
13	720
21	5040
34	40320
55	362880
```

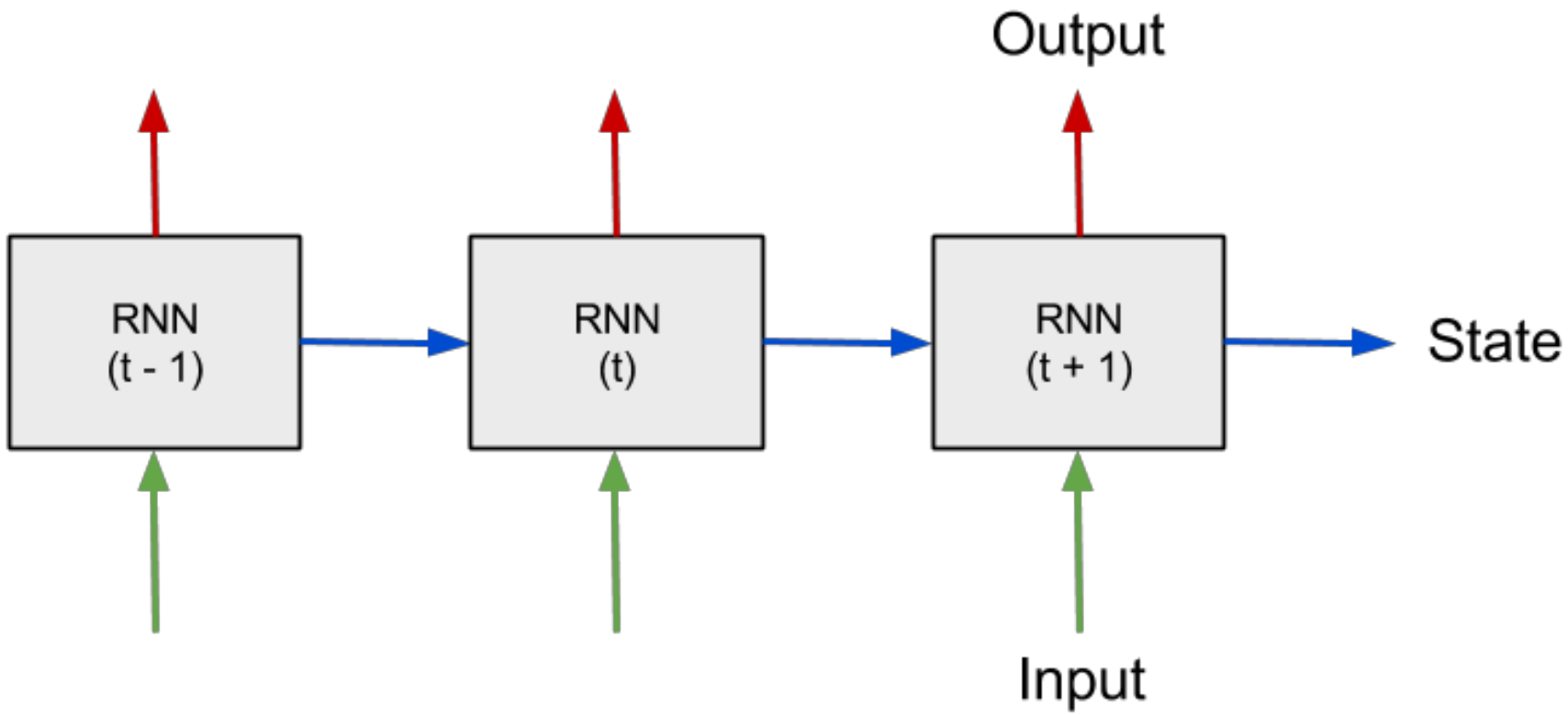
The Dependency in Neural Networks

Previous we tree the neural networks as：

Now We treat the neural networks as:
Elman network:

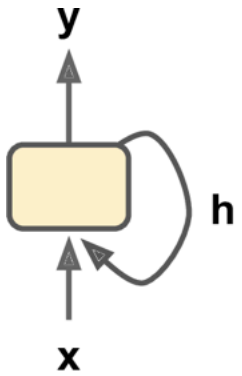
Or
Jordan networks:

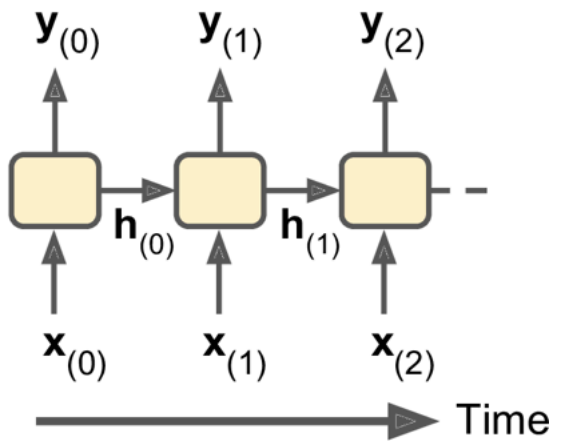
RNN (Recurrent Neural Networks)



RNNs Block

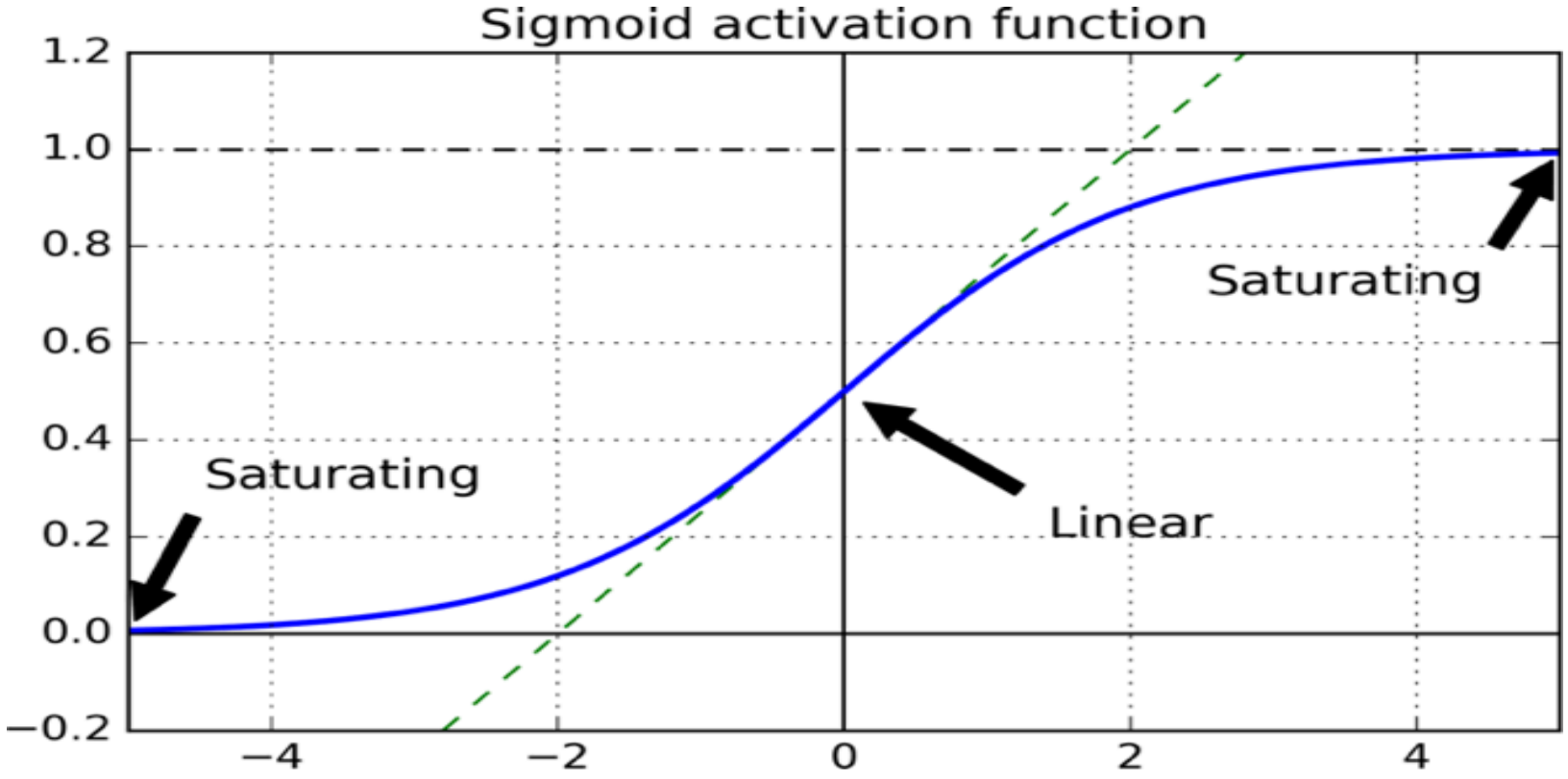
- From the input to the hidden state
- From the previous state to the next hidden st
- From the hidden state to the output.
- Stacked RNN
- Bidirectional RNN
- * RNN in tensorflow, in Keras.



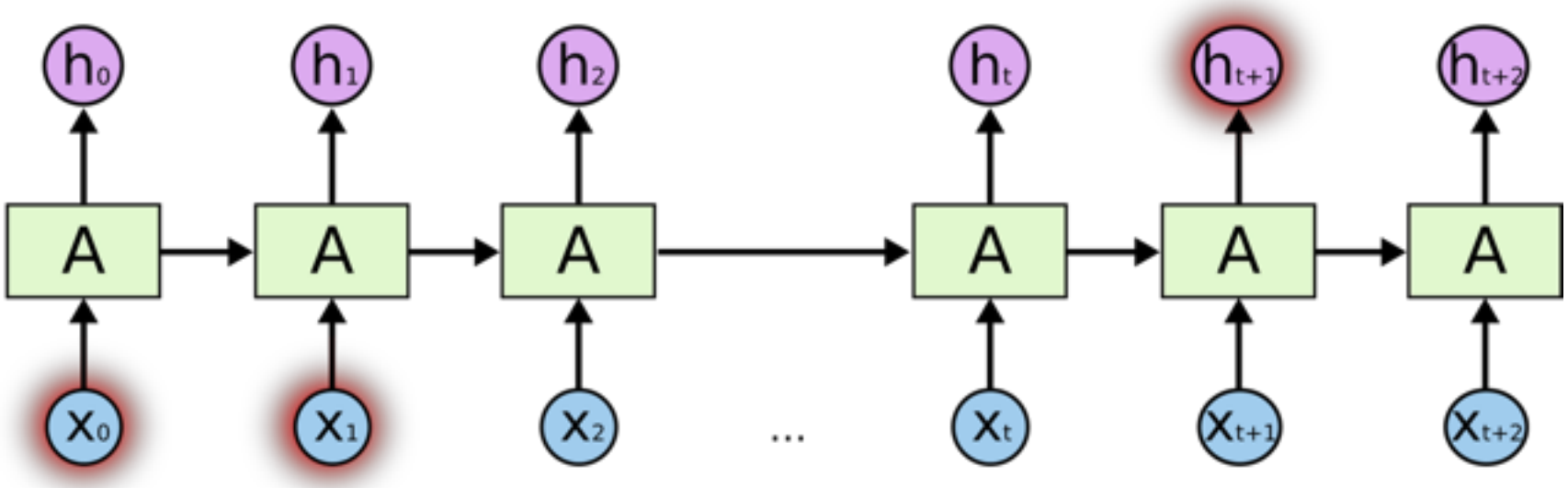


The Challenge of Long-Term Dependencies

- Vanishing, Exploding Problem
- Image climb mountains

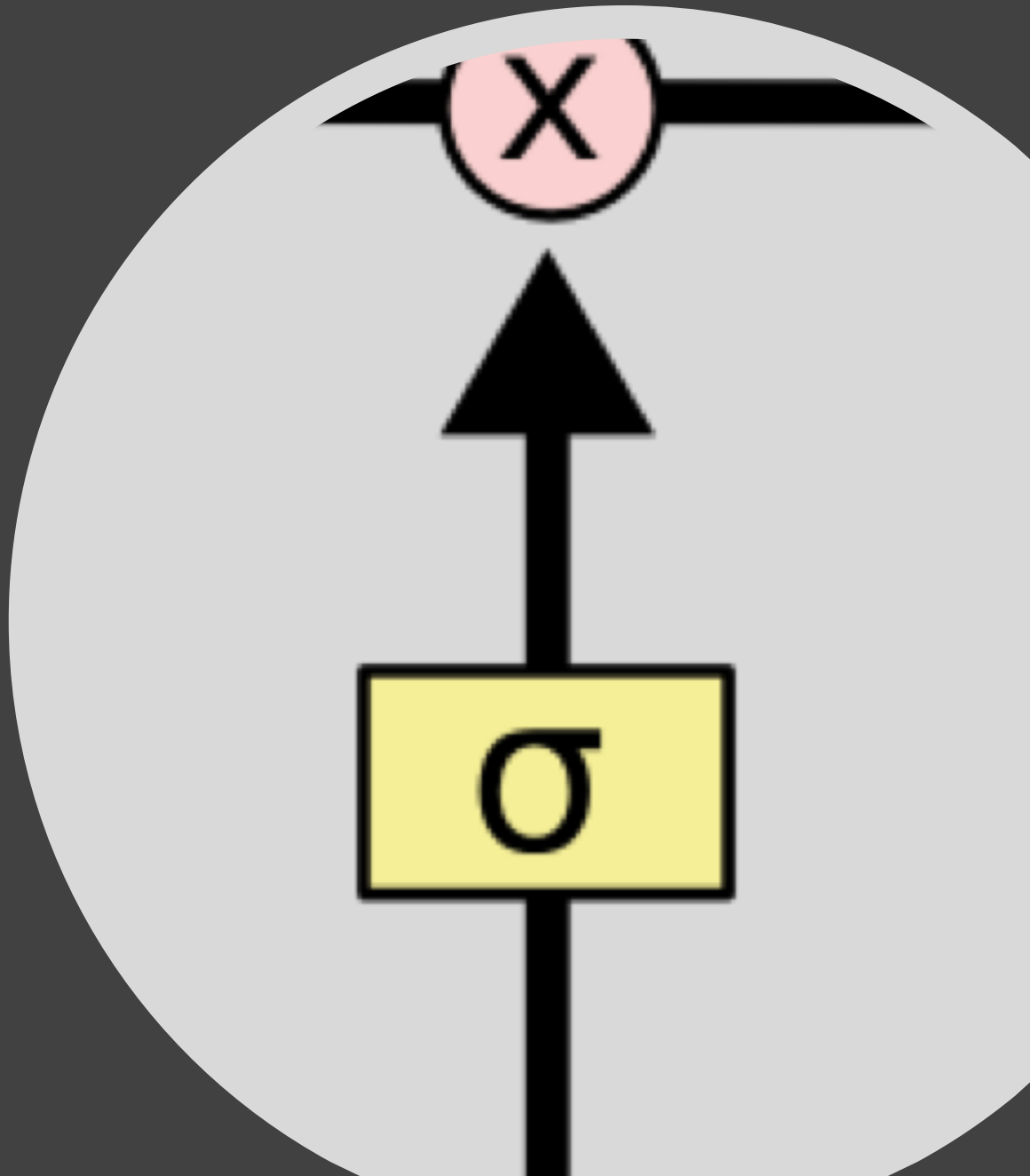


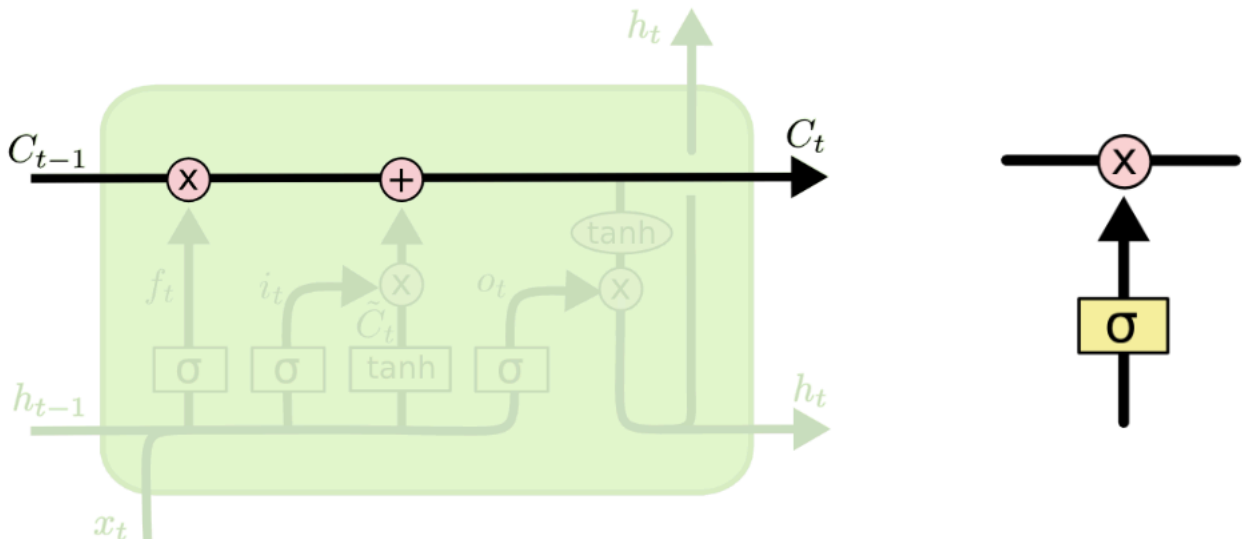
- E.g “I grew up in France… I speak fluent *French*.”



LSTM

- L-Long S-Short T-Term M-Memory
- Gate: Information Control
  - converge faster
  - detect long-term dependencies
  - Gates are a way to optionally let information through. They are composed out of **sigmoid** neural net layer and a pointwise multiplication operation.
  - Q: Why sigmoid?

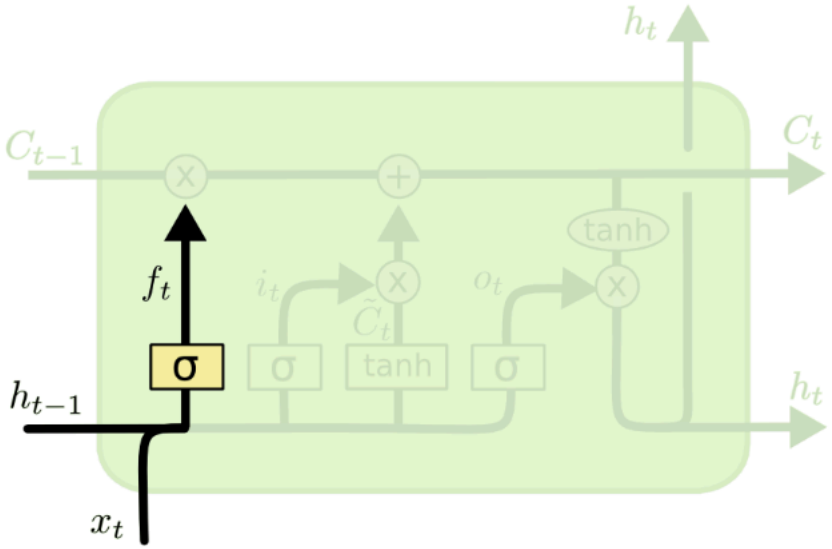



h_t
C_{t-1}
C_t
tanh
f_t
i_t
o_t
$\tilde{C}_t$
σ
σ
tanh
σ
h_{t-1}
h_t
x_t

σ

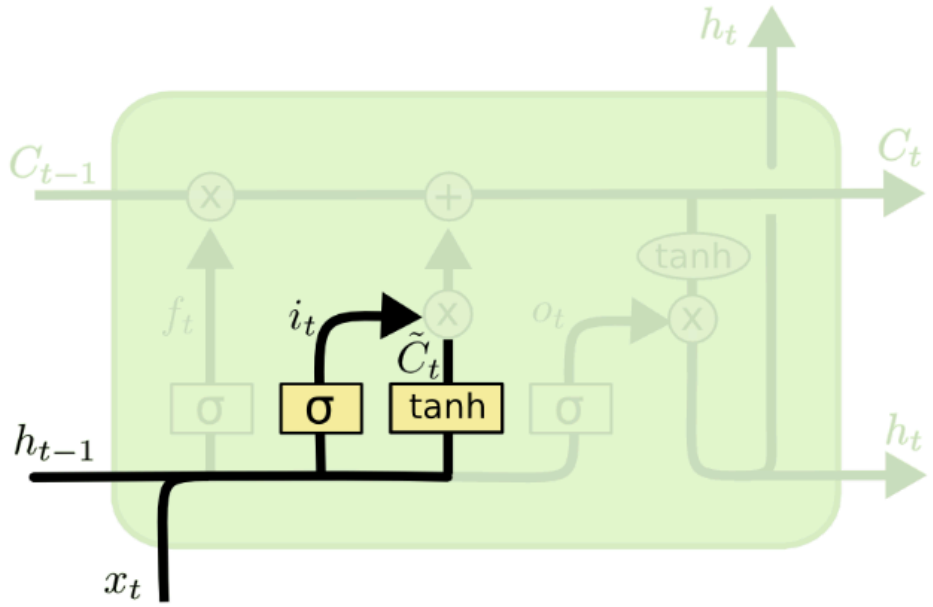
Forget Gate

decide what information we're going to throw away.



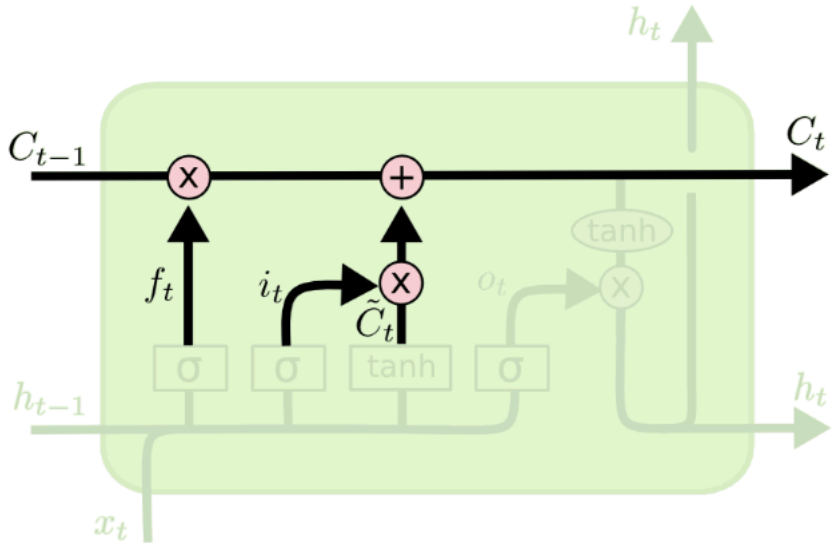
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right)$$
$$i_t = \sigma(W_t \cdot [h_{t-1}, x_t] + b_i)$$
$$\widetilde{C}_t = \tanh(W_c \cdot [h_{t-1}, x_t) + b_c\ ])$$
$$C_t = f_t\ * C_{t-1} + i_t\ * \widetilde{C}_t$$
$$\mathrm{O}_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$$
$$h_t = O_t\ * \tanh(C_t)$$

Input Gate



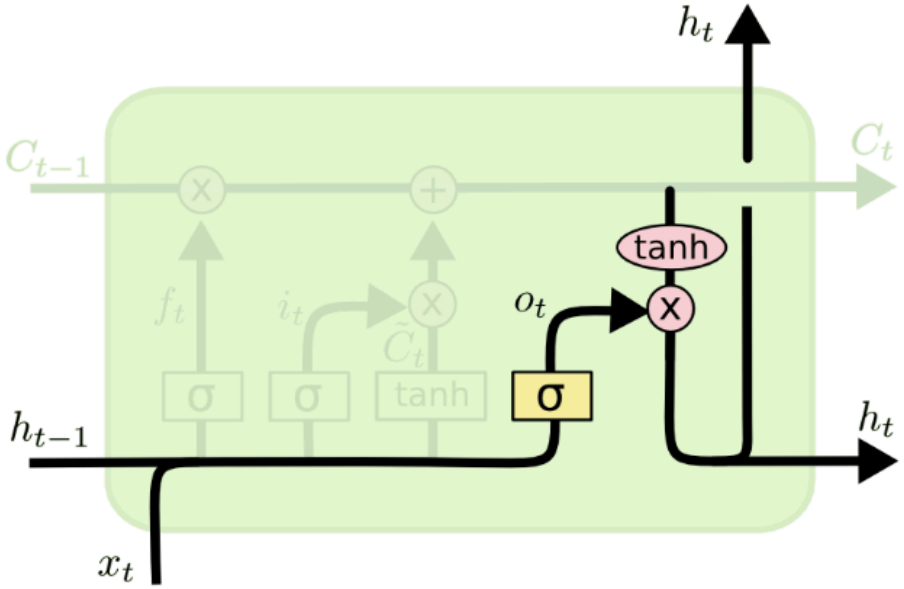
$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$
$$i_t = \sigma(W_t \cdot [h_{t-1}, x_t] + b_i)$$
$$\widetilde{C}_t = tanh(W_c \cdot [h_{t-1}, x_t) + b_c\ ])$$
$$C_t = f_t\ * C_{t-1} + i_t\ * \widetilde{C}_t$$
$$\mathrm{O}_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$$
$$h_t = O_t\ * \tanh(C_t)$$

Cell State

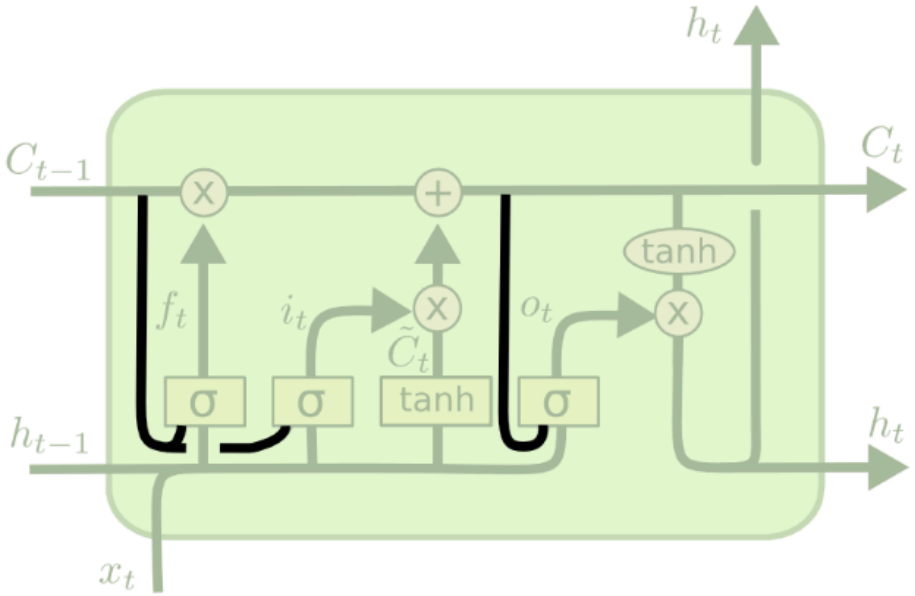


$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$
$$i_t = \sigma(W_t \cdot [h_{t-1}, x_t] + b_i)$$
$$\widetilde{C}_t = \tanh(W_c \cdot [h_{t-1}, x_t) + b_c\ ])$$
$$C_t = f_t\ * C_{t-1} + i_t\ * \widetilde{C}_t$$
$$O_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$$
$$h_t = O_t\ * \tanh(C_t)$$

Output Gate

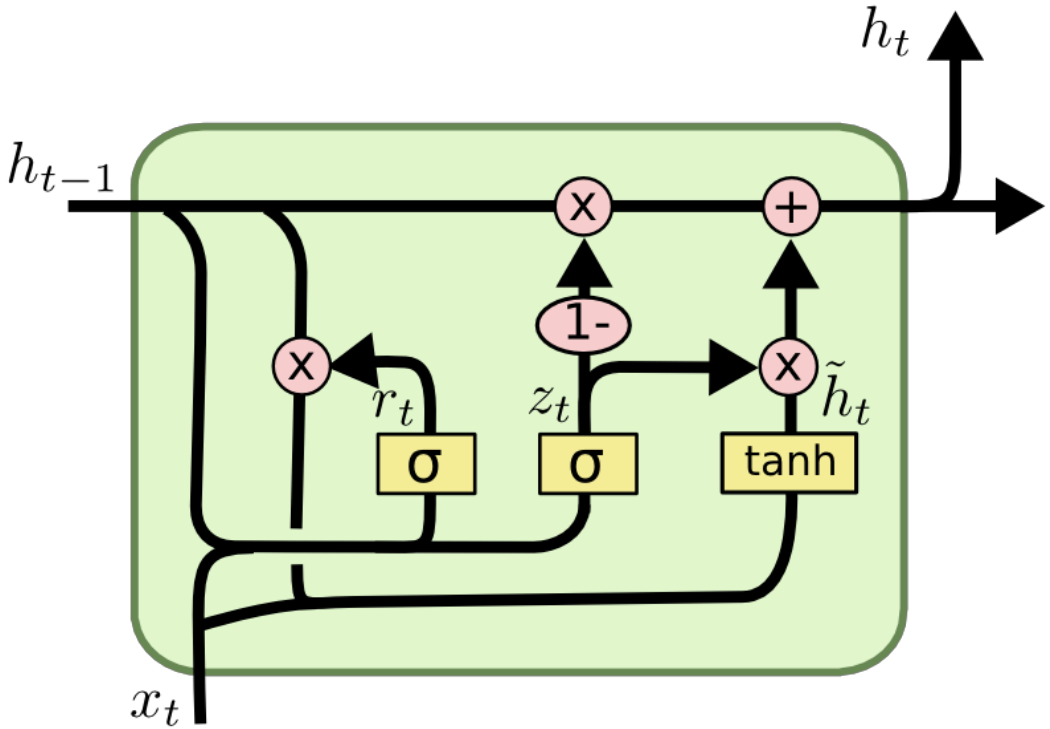


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right)$$
$$i_t = \sigma(W_t \cdot [h_{t-1}, x_t] + b_i)$$
$$\tilde{C}_t = \tanh(W_c \cdot [h_{t-1}, x_t) + b_c\ ])$$
$$C_t = f_t\ * C_{t-1} + i_t\ * \tilde{C}_t$$
$$O_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$$
$$h_t = O_t\ * \tanh(C_t)$$



$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_i\right)$$
$$o_t = \sigma\left(W_o \cdot [\boldsymbol{C_t}, h_{t-1}, x_t] \; + \; b_o\right)$$

Peephole connection
We let the gate layers look at the cell state



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

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

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

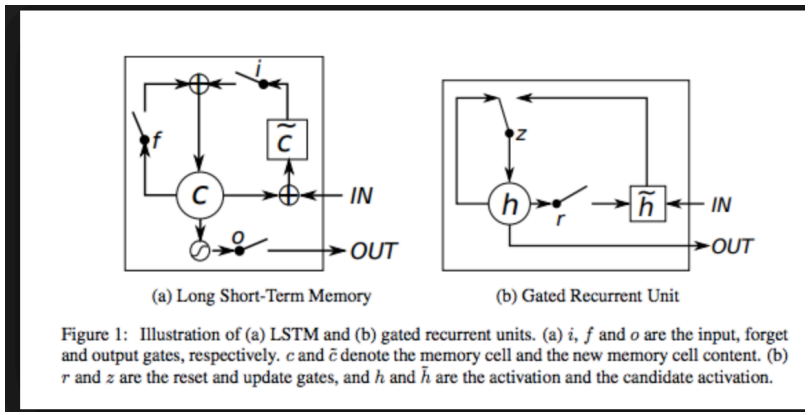
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Simpler and fewer parameters!

GRU: Gated Recurrent Unit

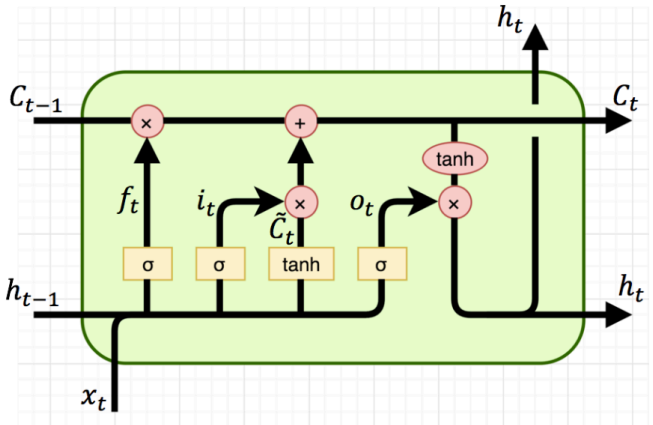
http://arxiv.org/pdf/1406.1078v3.pdf

- $f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right)$
- $i_t = \sigma(W_t \cdot [h_{t-1}, x_t] + b_i)$
- $\widetilde{C}_t = \tanh(W_c \cdot [h_{t-1}, x_t) + b_c\ ])$
- $C_t = f_t\ * C_{t-1} + i_t\ * \widetilde{C}_t$
- $\mathrm{O}_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$
- $h_t = O_t\ * \tanh(C_t)$



(a) Long Short-Term Memory (b) Gated Recurrent Unit

Figure 1: Illustration of (a) LSTM and (b) gated recurrent units. (a) i, f and o are the input, forget and output gates, respectively. c and $\tilde{c}$ denote the memory cell and the new memory cell content. (b) r and z are the reset and update gates, and h and $\tilde{h}$ are the activation and the candidate activation.



(a) Long Short-Term Memory

Assignment:

- Reading:
  - 1. Hands-on tensorflow and scikit learning
  - 2. https://colah.github.io/posts/2015-08-Understanding-LSTMs/
  - 3. https://arxiv.org/pdf/1506.00019.pdf
  - 4. https://arxiv.org/pdf/1409.3215.pdf
  - 5. https://www.kaggle.com/suniliitb96/tutorial-keras-transfer-learning-with-resnet50/notebook
- Kaggle Project: https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge
  - 找到一个notebook解决方案，修改源代码，使你的结果超过原始作者
  - 然后使用修改过的解决方法，解决豆瓣评分的问题
  - 把自己的源代码都发布到GitHub上