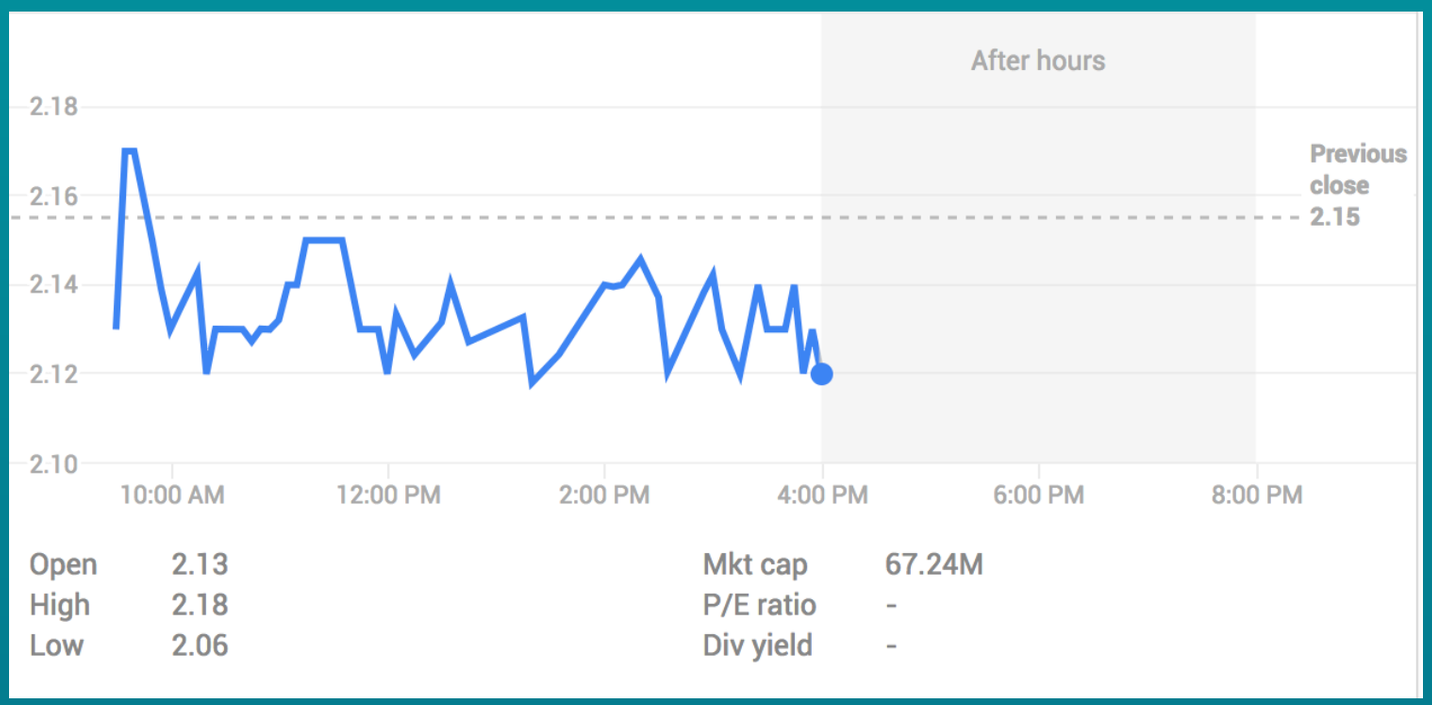
Outlines

- 0. Pre-coding exercise
- 1. 句子向量 Sentence Embedding;
- — 莫智文 + 阿楠 (share speech)
- 2. RNN 神经网络的初步

RNN的提出背景及基本原理

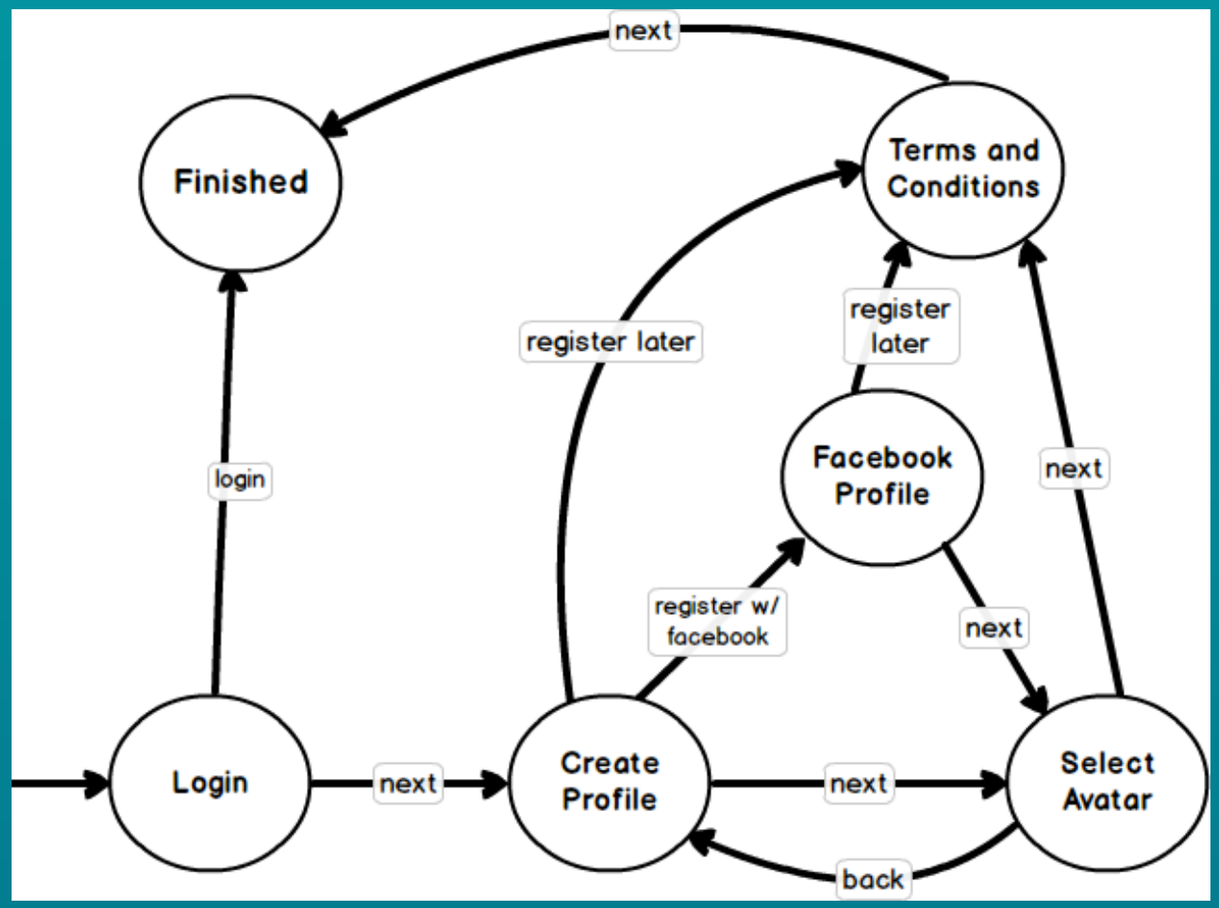
RNN 的提出背景

- 1. 序列问题：
  - 股市， 文本， 用户行为；



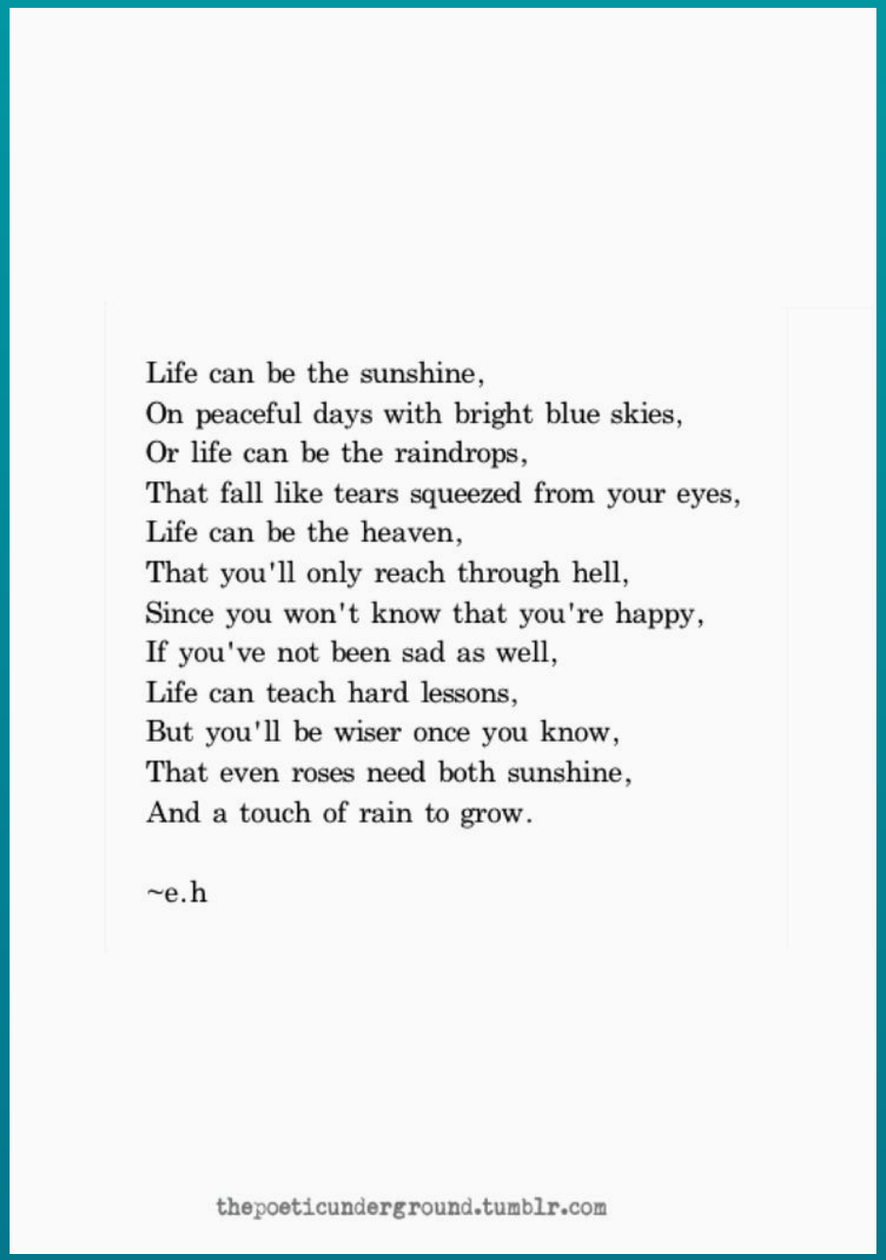
RNN 的提出背景

- 2. 时序状态：
  -



RNN 的提出背景

- 3. 变长输入:



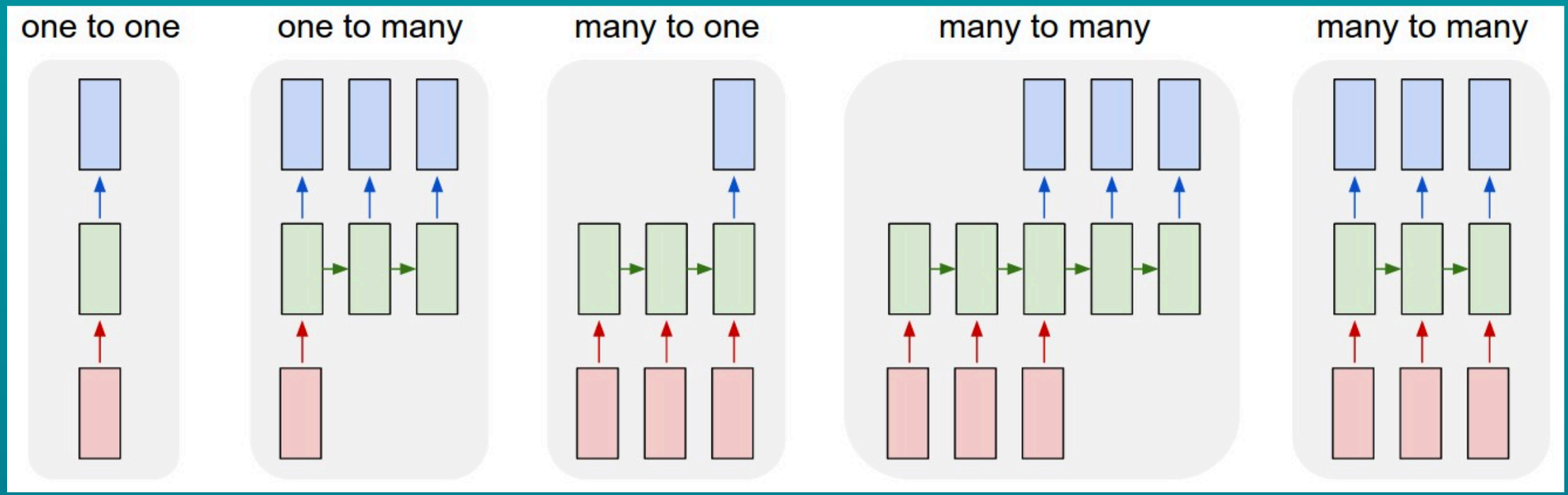
Life can be the sunshine,
On peaceful days with bright blue skies,
Or life can be the raindrops,
That fall like tears squeezed from your eyes,
Life can be the heaven,
That you'll only reach through hell,
Since you won't know that you're happy,
If you've not been sad as well,
Life can teach hard lessons,
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

序列问题

- 序列问题的不同类型



新的模型

- 1. 当前step的信息需要和上一个step相关；

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
```

新的模型

$$y_t = \sigma(Wx_t + b)$$
$$y_{t+1} = \sigma(Wx_{t+1} + b)$$

- 在神经网络里如何将前后两个信息进行相连?

- Elman network:

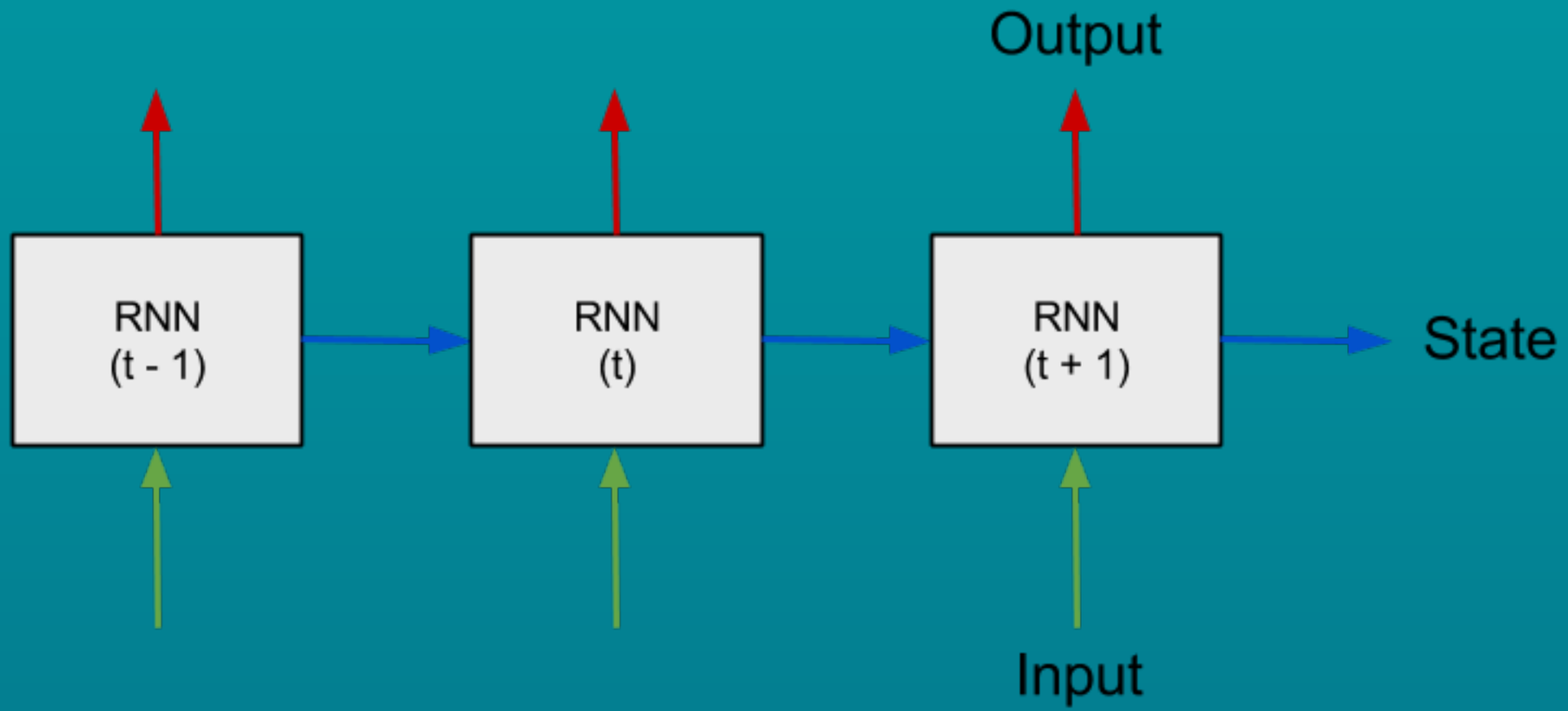
$$h_t = \sigma h(W_h x_t + U_h h_{t-1} + b_h)$$
$$y_t = \sigma_y(W_y h_t + b_y)$$

- Jordan network:

$$h_t = \sigma h(W_h x_t + U_h y_{t-1} + b_h)$$
$$y_t = \sigma_y(W_y h_t + b_y)$$

RNN模型

- RNN (Recurrent Neural Networks)



- 1. 下载并阅读论文： https://arxiv.org/pdf/1506.00019.pdf;
- 2. Hands on Tensorflow P.379 - P410
- Next: LSTM, GRU

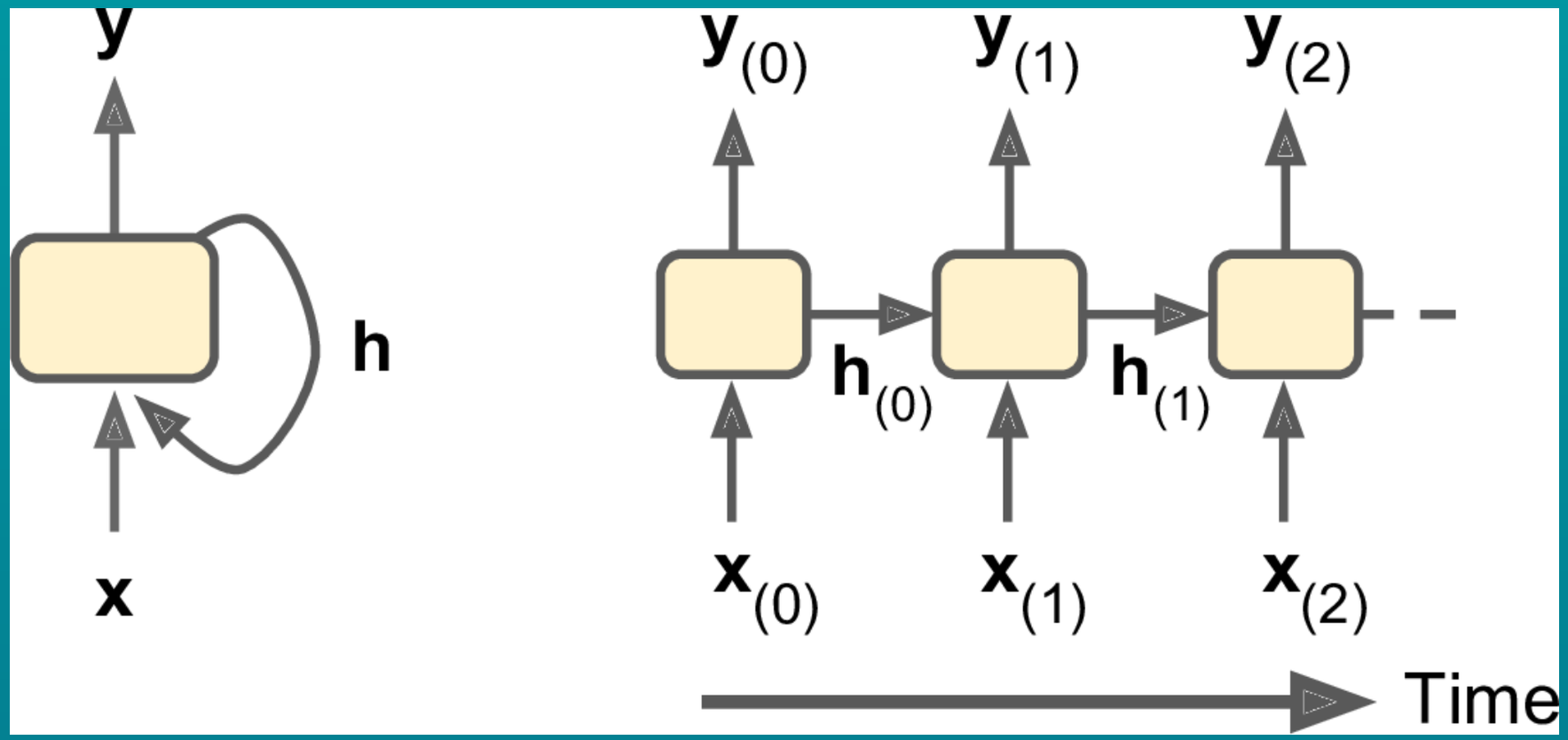
RNN的变体LSTM, GRU

RNNs Block

$$h_t = \sigma h(W_h x_t + U_h h_{t-1} + b_h)$$
$$y_t = \sigma_y(W_y h_t + b_y)$$

- From the input to the hidden state;
- From the previous state to the next hidden state;
- From the hidden state to the output;

RNN 的处理单元



The Challenge of Long-Term Dependencies

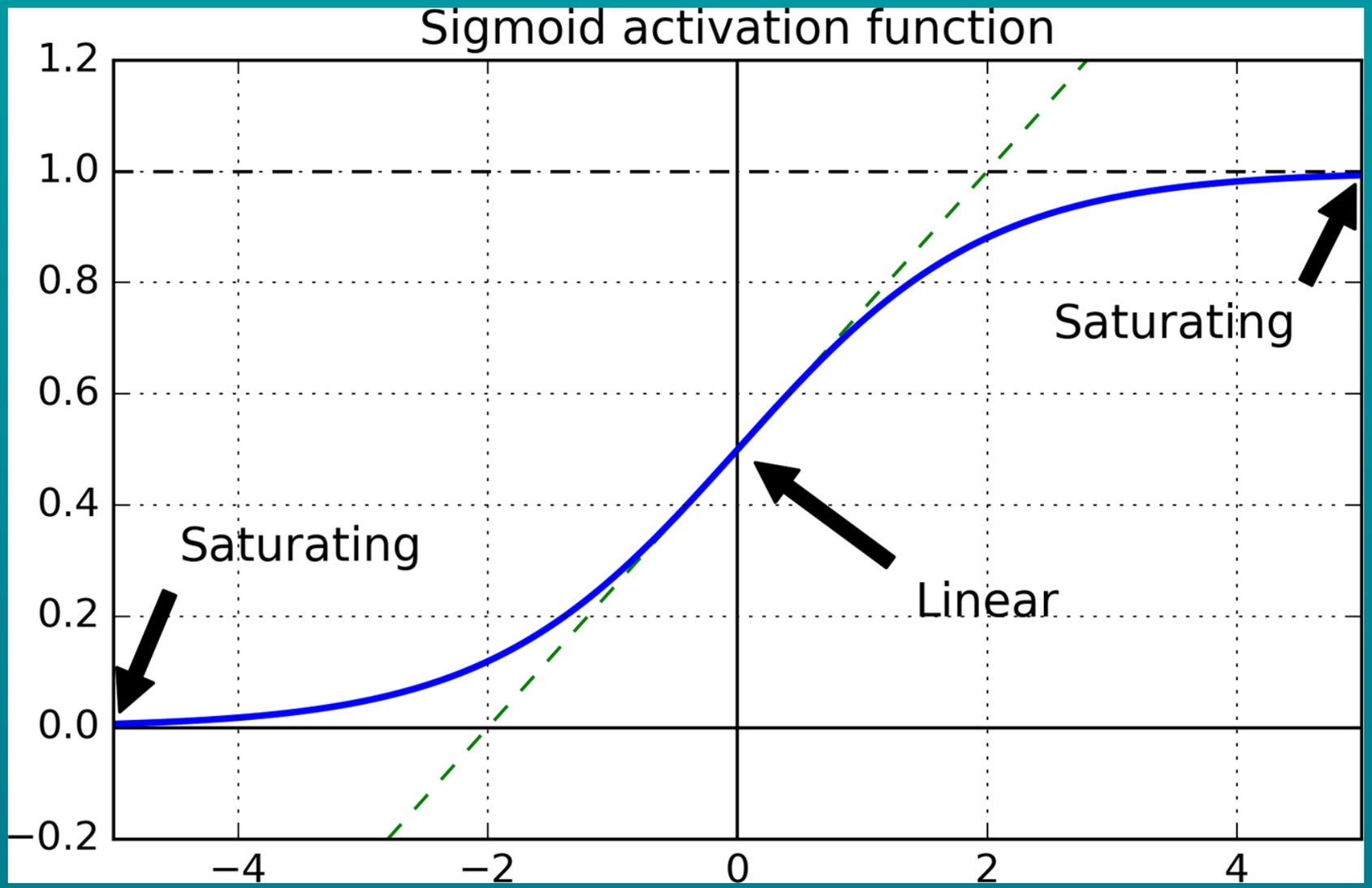
- Vanishing Problem：（消失，消散）

- 神经网络的更新依靠梯度进行；

- 该训练step, 梯度越大，参数调整越大

- 实际中，梯度会越来越小，就像人下山；

非线性函数的梯度


Sigmoid activation function
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−4
−2
0
2
4
Saturating
Saturating
Linear

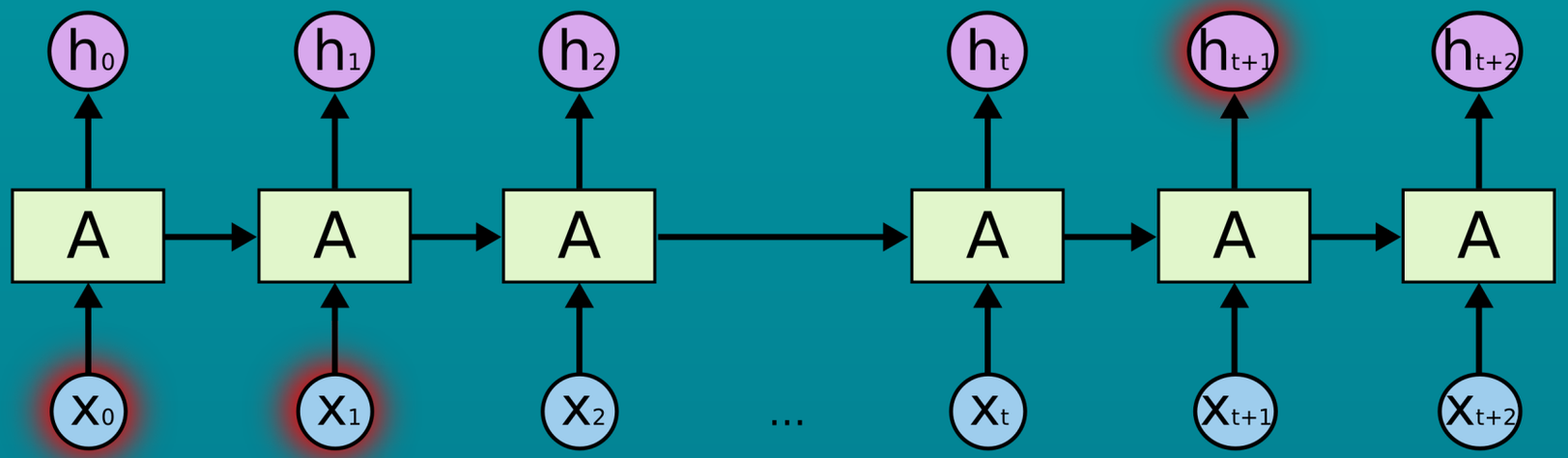
The Challenge of Long-Term Dependencies

- Vanishing 的另外一个原因：
  - 不断的非线性的叠加；

- exploding: 很多步训练中， 偏导的不断叠加， 使得早期步数参数的梯度可能会变得非常大；

长信息和短信息的同时获取

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



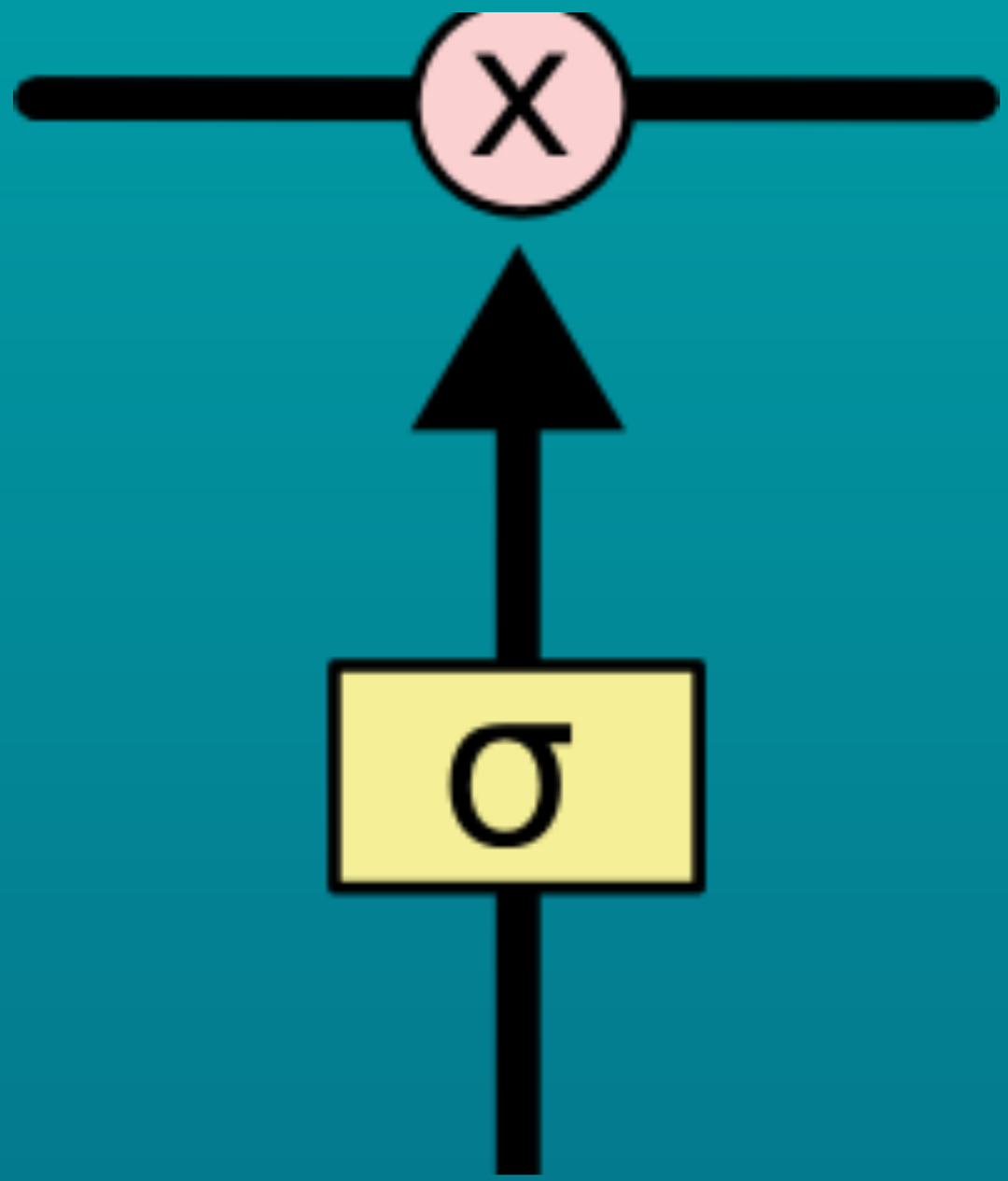
LSTM

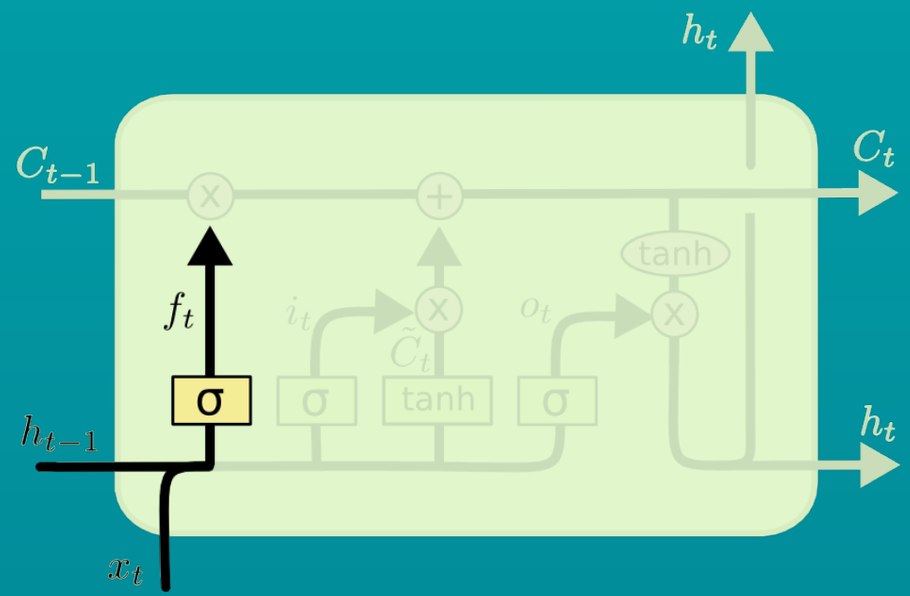
- L-Long S-Short T-Term M-Memory
- 核心思想： 阀门机制， 选择性忽略某些信息
  - converge faster(收敛更快)
  - detect long-term dependencies

Gate

- 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?

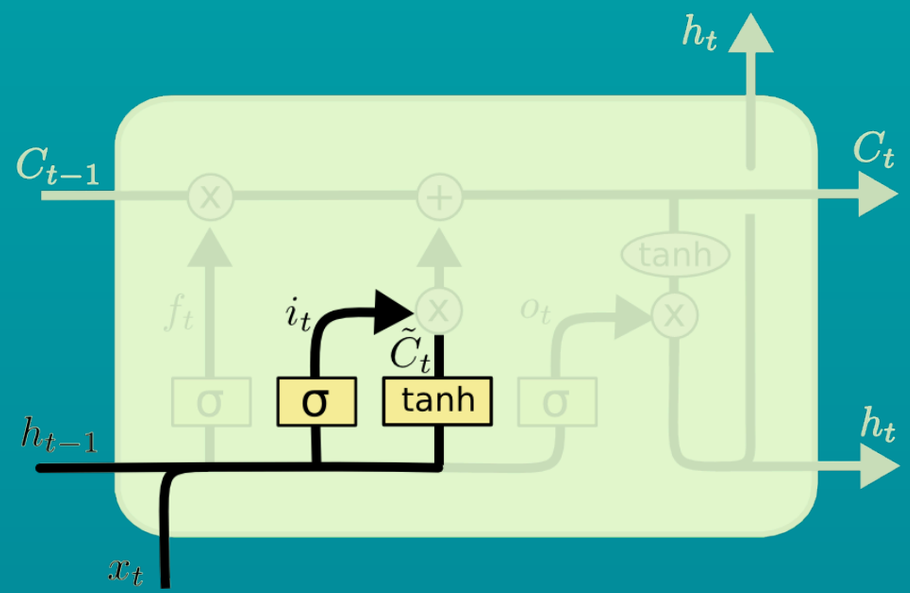




$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Forget Gate

decide what information we're going to throw away.

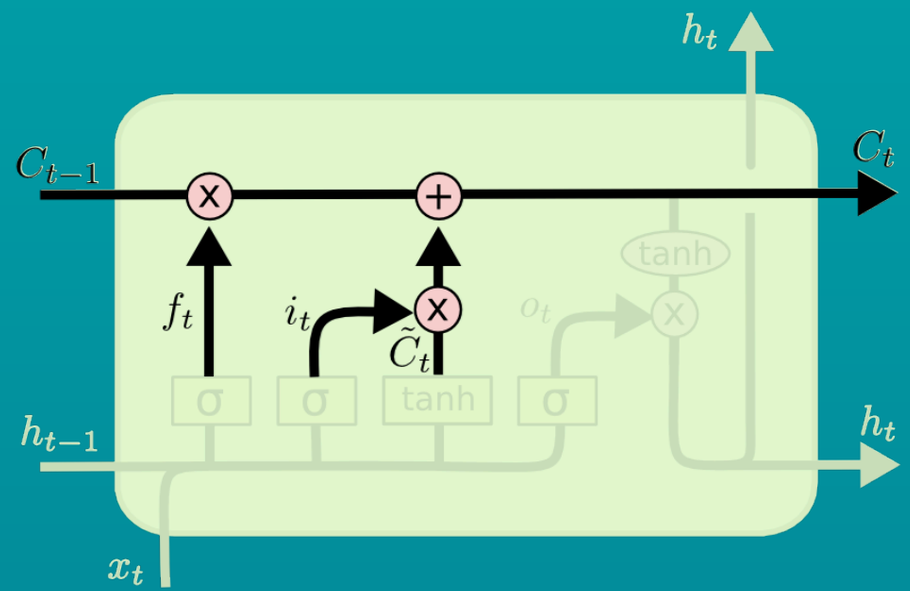


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

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

Input Gate

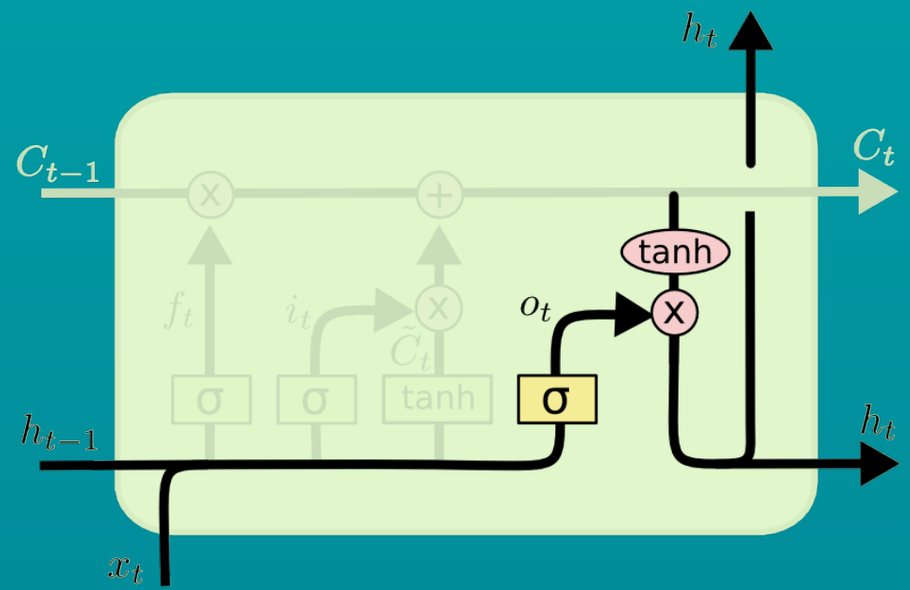
input gate: sigmoid layer decides which values we'll update;
candidate value: nolinear change for x.



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

Execute forget and input

New candidate values, scales by how much we decided to update each state value.

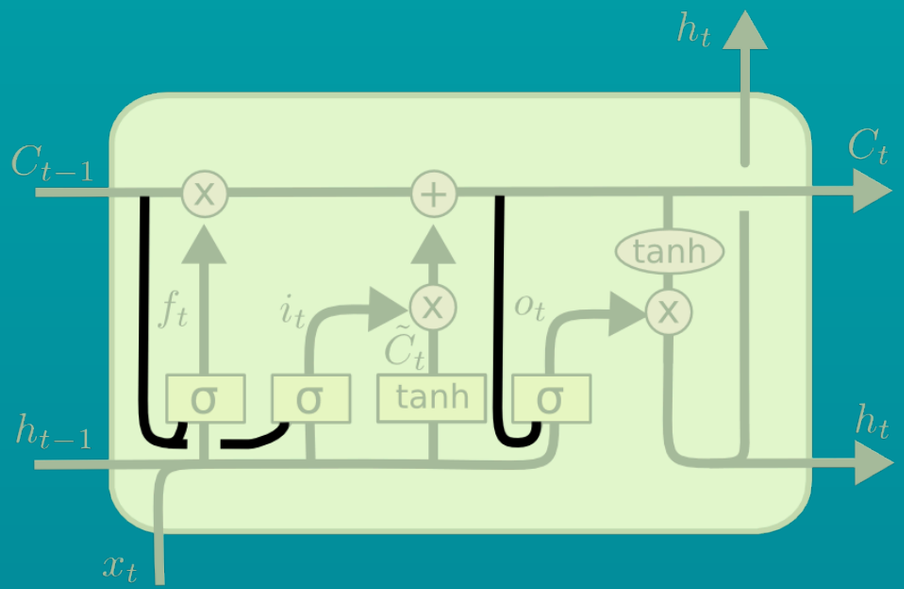


$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

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

Output Gate

output gate decides how much we are going to output. Then, we put the cells state through tanh, so that we only output the parts we decides to.



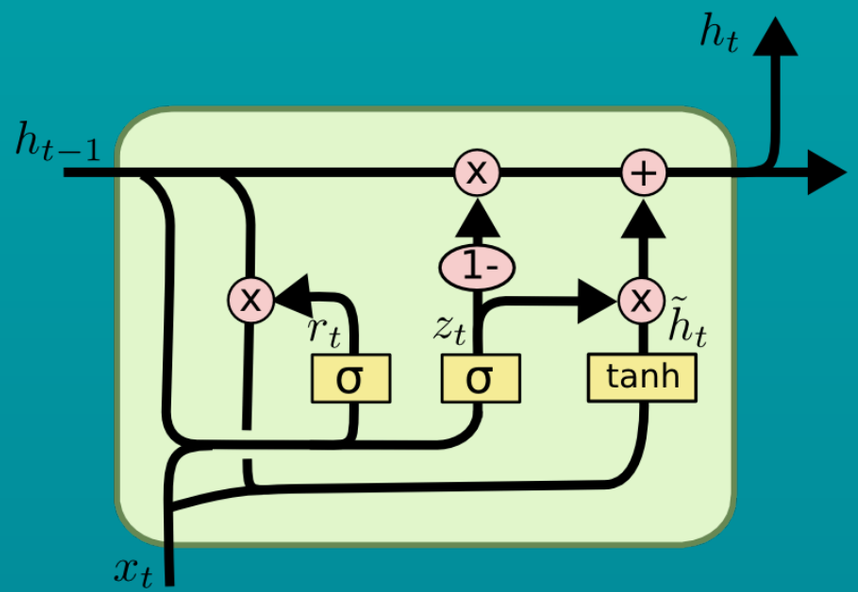
$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_i\right)$$
$$o_t = \sigma\left(W_o \cdot [\boldsymbol{C_t}, h_{t-1}, x_t] \; + \; b_o\right)$$

LSTM Variants

peehole connection

GRU

- Combine the forget and input gates into a single ‘update gate’;
- Merge the cell state and hidden state;
- No output gate.



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

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

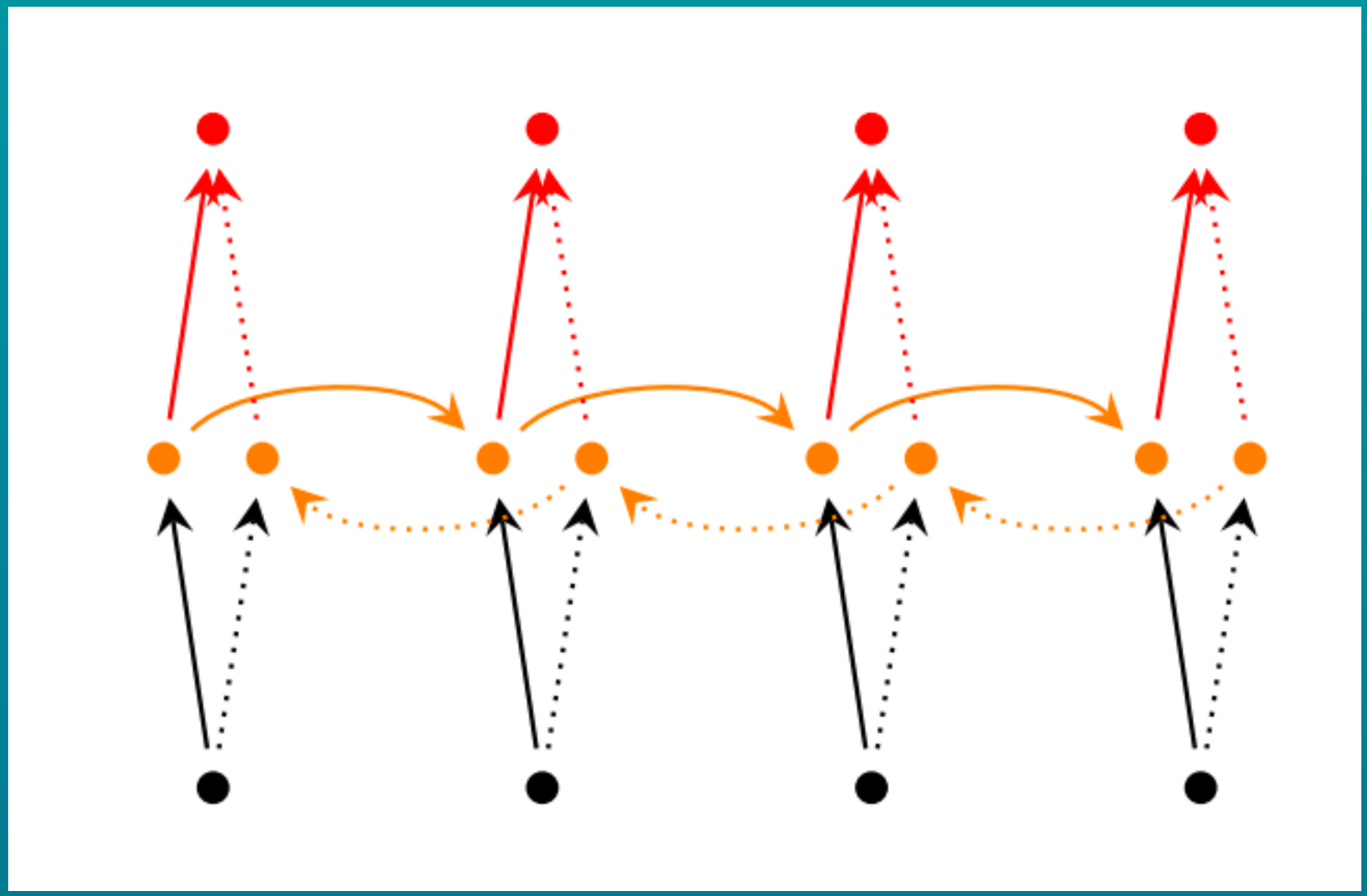
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

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

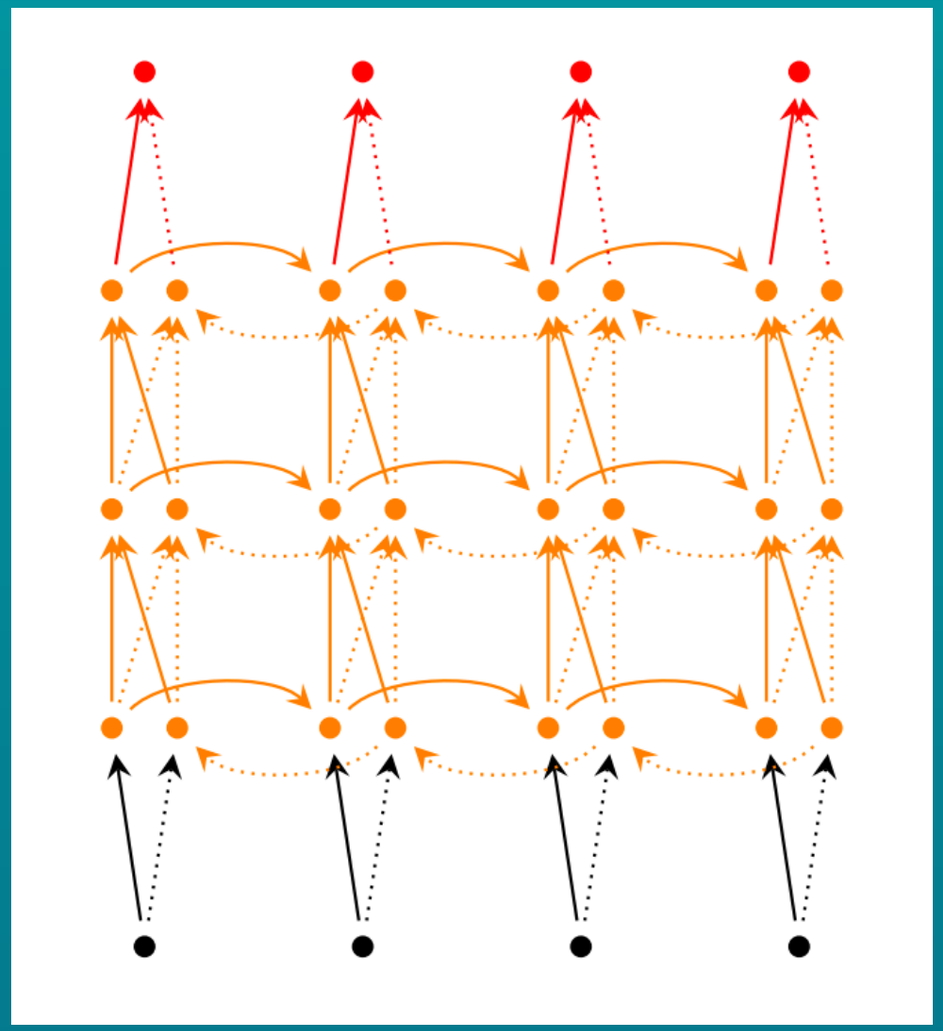
GRU

其他变体

- Bidirectional RNN



- Stacked RNNs

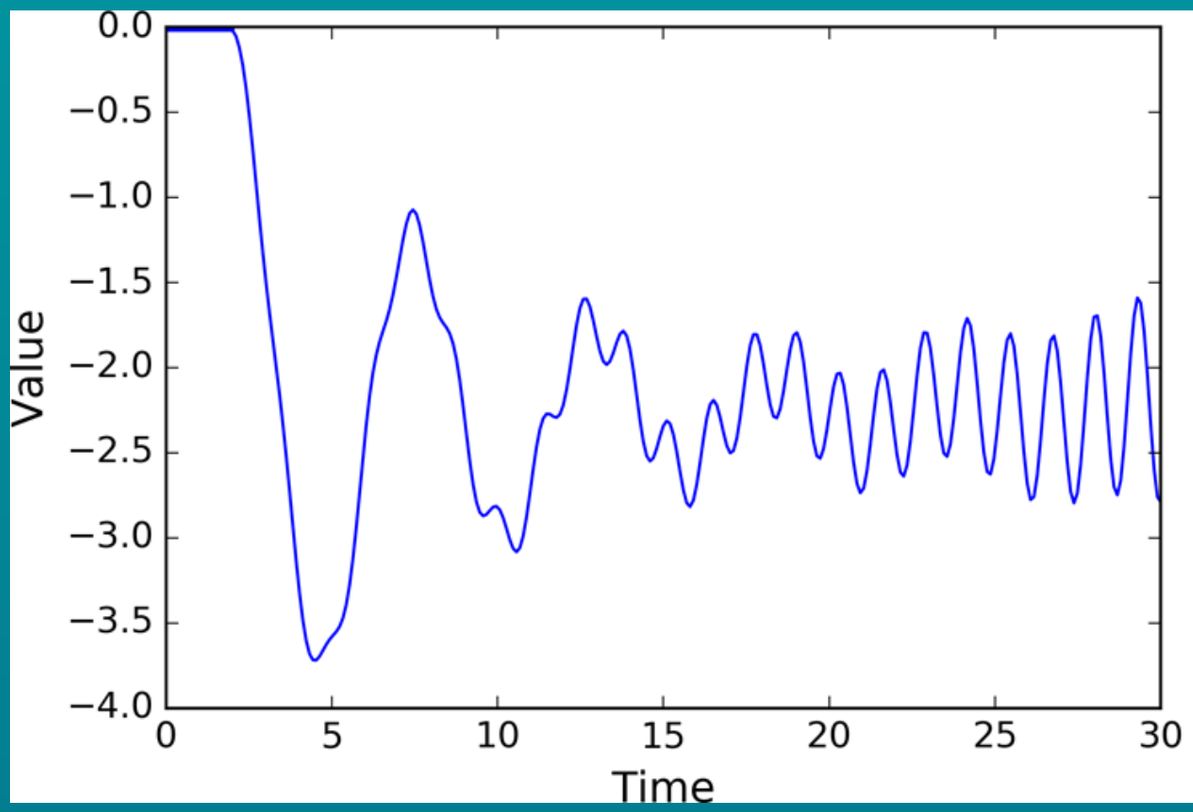


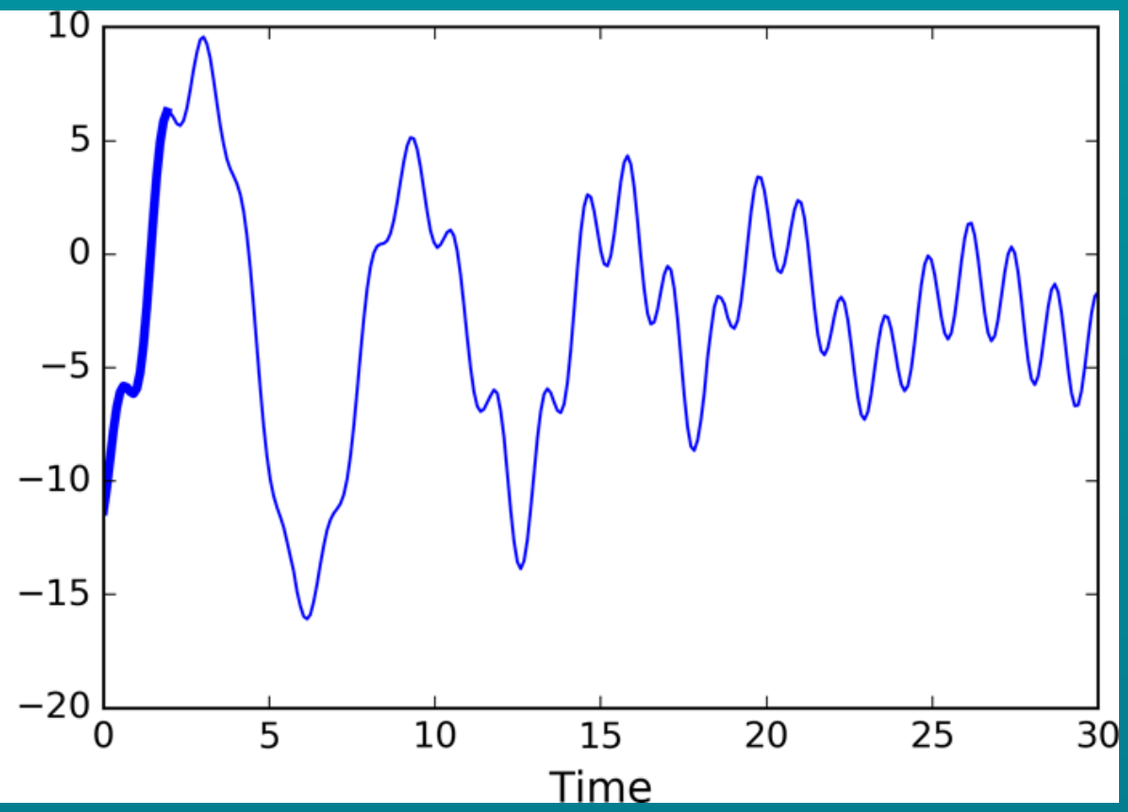
TensorFlow & RNN

- BasicRNNCell
- MultiRNNCell
- LSTM, GRU

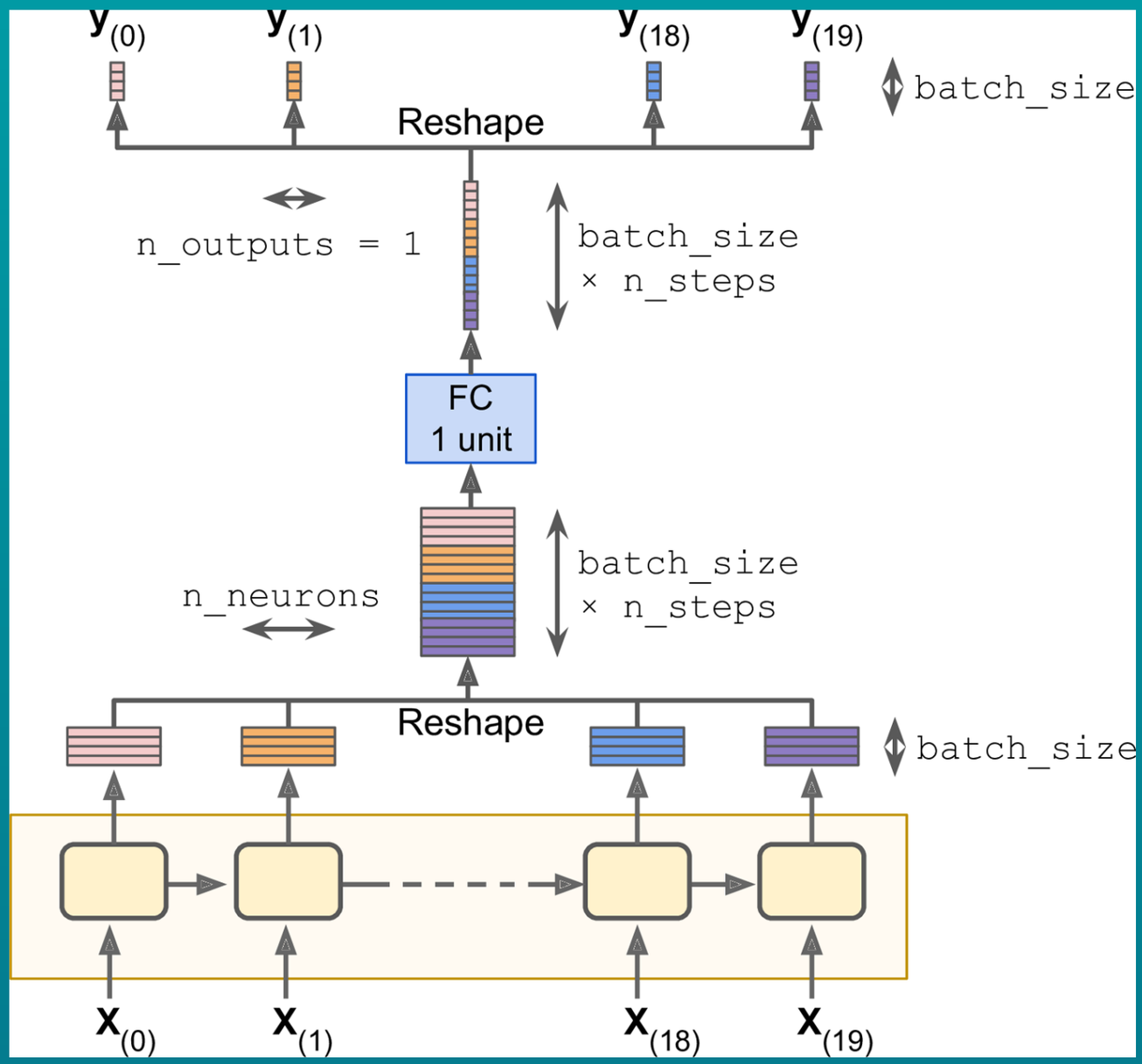
RNN 进行预测

- 依据某一序列的输入， 预测该序列的下一个元素





Reshape


$y_{(0)}$
$y_{(1)}$
$y_{(18)}$
$y_{(19)}$
batch_size
Reshape
n_outputs = 1
batch_size
× n_steps
FC
1 unit
n_neurons
batch_size
× n_steps
Reshape
batch_size
$x_{(0)}$
$x_{(1)}$
$x_{(18)}$
$x_{(19)}$

- 1. Predicate RNN
- 2. Stacked or MultiRNNs
- 3. LSTM & GRU
- 4. Pipeline RNN
- 5. Dropout

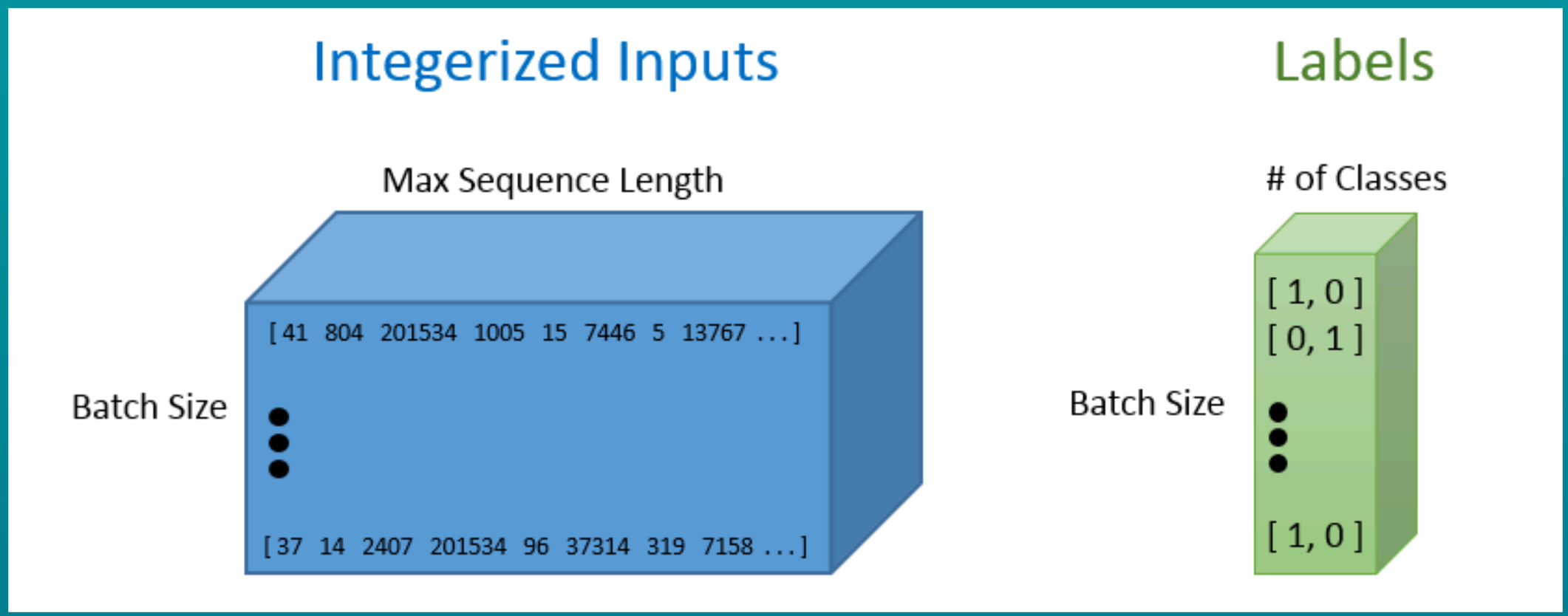
RNN进行文本分类实例

Sentence Classification

- 任务分类;
- 情感分类;

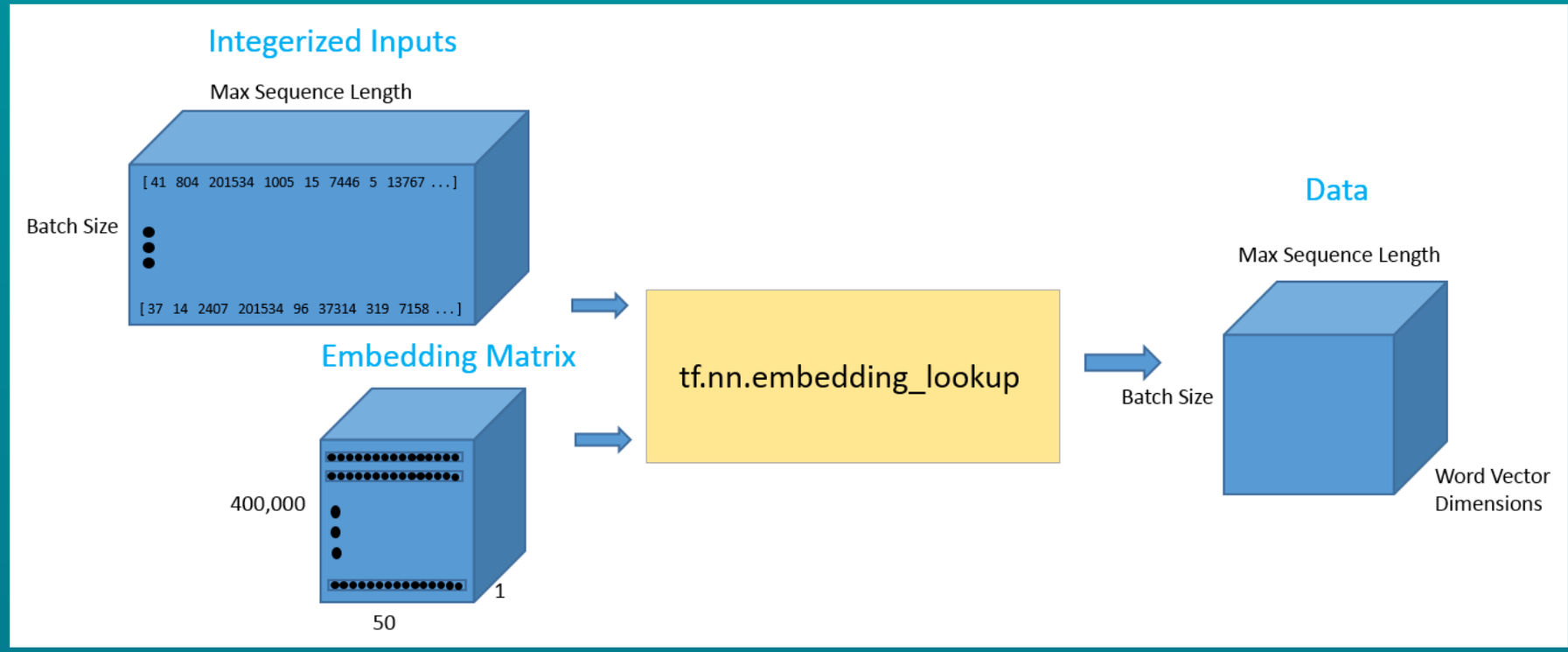
Descript	Category
GRAND THEFT FROM LOCKED AUTO	LARCENY/THEFT
POSSESSION OF NARCOTICS PARAPHERNALIA	DRUG/NARCOTIC
AIDED CASE, MENTAL DISTURBED	NON-CRIMINAL
AGGRAVATED ASSAULT WITH BODILY FORCE	ASSAULT
ATTEMPTED ROBBERY ON THE STREET WITH A GUN	ROBBERY

用tensorflow实现情感分类



用tensorflow实现情感分类

- embedding_lookup: 依据序号获取某个单词对于的word2vec.



流程

- 1. 将单词切词:
    - jieba.cut
- 2. 单词变为index:
    - 自定义实现或tensorflow textprocessing
    - id_to_word = {}; word_to_id = {}
- 3. padding；
- 4. batchized
- 5. training;

代码分析

- Keras Example
- https://github.com/tensorflow/tensorflow/blob/master/tensorflow/examples/udacity/6_lstm.ipynb
- https://keras.io/getting-started/sequential-model-guide/#examples

RNN进行实体识别

Name Entity Recognition

- 特朗普在华盛顿对CNN记者很生气。

- 特朗普[*person*]在华盛顿[*location*]对CNN[*organization*]记者很生气。

- From wikipedia:

Named-entity recognition (**NER**) (also known as **entity identification**, **entity chunking** and **entity extraction**) is a subtask of information extraction that seeks to locate and classify named entities in text into pre-defined categories such as the names of persons, organizations, locations, expressions of times, quantities, monetary values, percentages, etc.

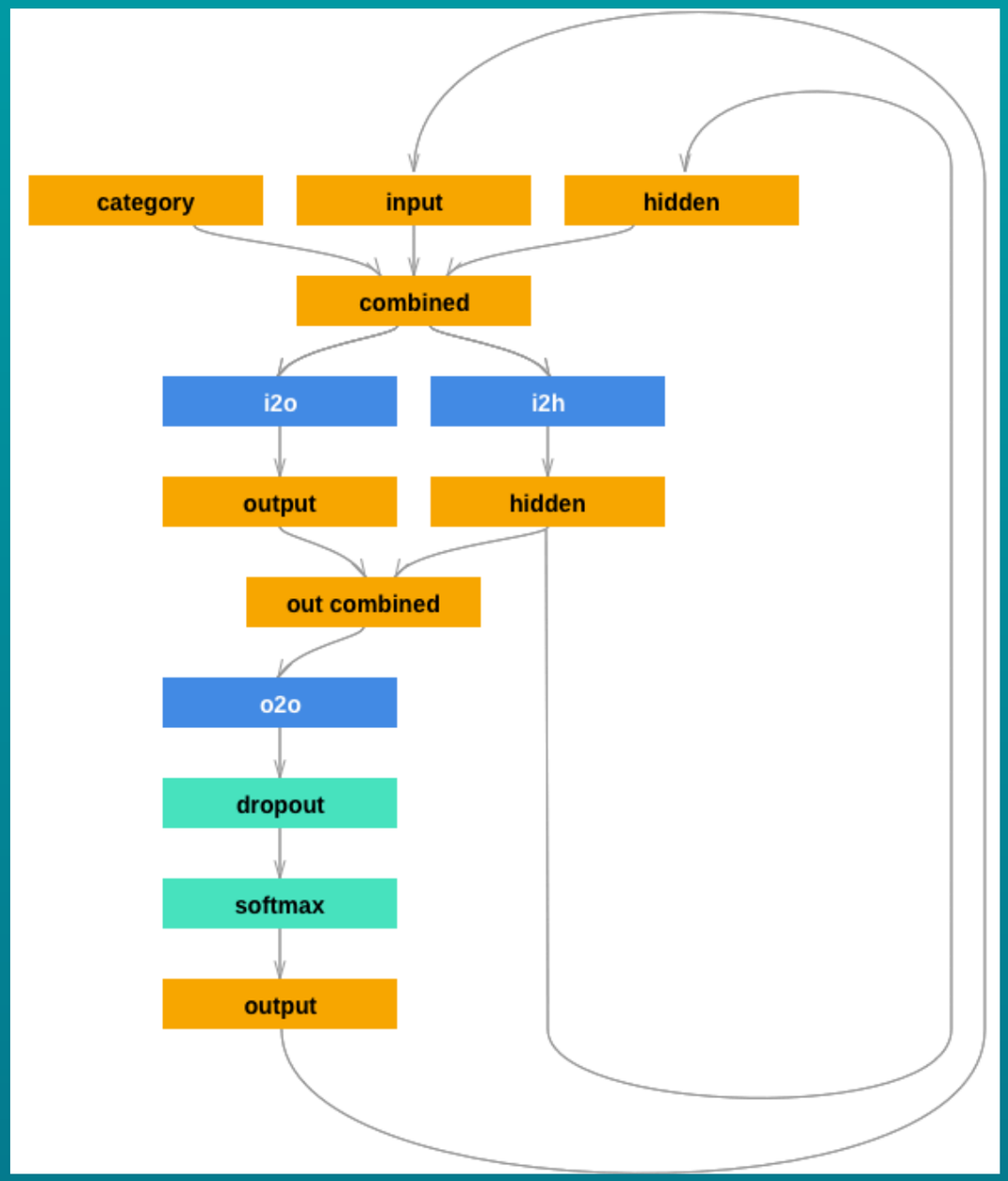
其他实体识别的方法

- jieba 分词： postag
- Stanford CORENLP: 直接调用即可
- HIT PyLTP: 直接调用即可, 只支持python 2
- Next: RNN TensorFlow Code

使用RNN生成文本

生成模型的特点

- 1. 前一个时刻的输出是后一个时刻的输入；
- 2. 每次向前一步， 寻找最优的输出



- 迭代输入：

```
criterion = nn.NLLLoss()

learning_rate = 0.0005

def train(category_tensor, input_line_tensor, target_line_tensor):
    hidden = rnn.initHidden()

    rnn.zero_grad()

    loss = 0

    for i in range(input_line_tensor.size()[0]):
        output, hidden = rnn(category_tensor, input_line_tensor[i], hidden)
        loss += criterion(output, target_line_tensor[i])

    loss.backward()

    for p in rnn.parameters():
        p.data.add_(-learning_rate, p.grad.data)

    return output, loss.data[0] / input_line_tensor.size()[0]
```

如何进行预测

- 利用以及训练好的参数

```
max_length = 20

# Sample from a category and starting letter
def sample(category, start_letter='A'):
    category_tensor = Variable(categoryTensor(category))
    input = Variable(inputTensor(start_letter))
    hidden = rnn.initHidden()

    output_name = start_letter

    for i in range(max_length):
        output, hidden = rnn(category_tensor, input[0], hidden)
        topv, topi = output.data.topk(1)
        topi = topi[0][0]
        if topi == n_letters - 1:
            break
        else:
            letter = all_letters[topi]
            output_name += letter
        input = Variable(inputTensor(letter))

    return output_name
```

输出实例

- 1 只能指出短文本， 例如姓名生成；

- 2. 长文本具有的问题：

  - 1. 重复： 黄蓉说： 靖哥哥， 靖哥哥， .. , 靖哥哥

  - 2. 没有逻辑；

  - 3. 不能处理OOV问题

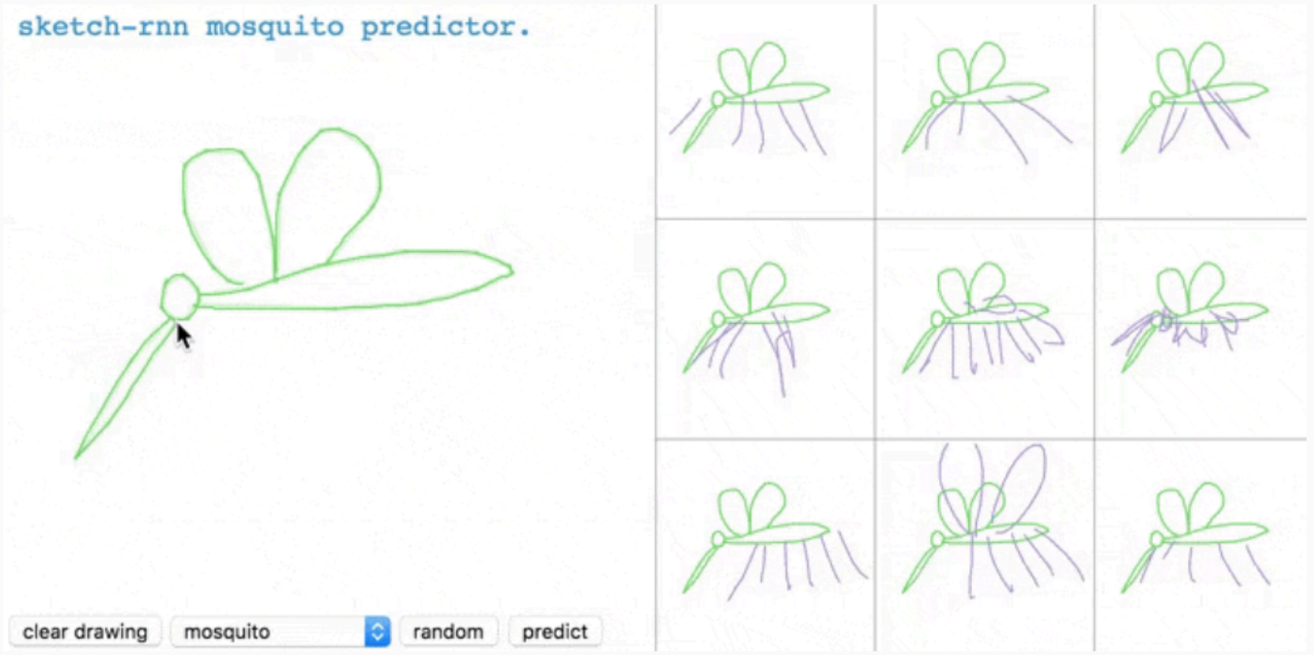
```
samples('Russian', 'RUS')

samples('German', 'GER')

samples('Spanish', 'SPA')

samples('Chinese', 'CHI')
```

Out:

```
Roukov
Uarinov
Sharan
Gerterrin
Erengen
Rouran
Sallan
Perra
Allanase
Cha
Han
Iun
```

扩展

- 1. 自动作曲；
- 2. 自动绘画： Google Auto Draw



Assignments

- 根据课程讲述， 用TensorFlow LSTM实现姓名生成的模型

Assignments

- 使用RNN\LSTM\GRU 网络结构，对豆瓣信息进行自动打分.